ESC-TR-95-001

Project Report
RASSP-1
(Rev. 1)

RASSP Benchmark 1 Technical Description

B.W. Zuerndorfer
J.C. Anderson
R.A. Ford
A.H. Anderson
G.A. Rocco
G.A. Shaw

25 January 1995

Lincoln Laboratory

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

LEXINGTON, MASSACHUSETTS

Prepared for the Advanced Research Projects Agency
under Air Force Contract F19628-95-C-0002.

This report is based on studies performed at Lincoln Laboratory, a center for research operated by Massachusetts Institute of Technology. The work was sponsored by the Advanced Research Projects Agency under Air Force Contract F19628-95-C-0002.

MASSACHUSETTS INSTITUTE OF TECHNOLOGY
LINCOLN LABORATORY

RASSP BENCHMARK 1 TECHNICAL DESCRIPTION

B.W. ZUERNDORFER
Group 93

J.C. ANDERSON
R.A. FORD
G.A. SHAW
Group 97

A.H. ANDERSON
Group 23

G.A. ROCCO
Group 42

Accesion For		
NTIS CRA&I		✓
DTIC TAB		☐
Unannounced		☐
Justification		
By		
Distribution /		
Availability Codes		
Dist	Avail and / or Special	
A-1		

PROJECT REPORT RASSP-1 (Rev. 1)

25 JANUARY 1995

LEXINGTON MASSACHUSETTS

ABSTRACT

This report describes the first in a series of application problems which are intended to measure the performance of a process for rapid prototyping of embedded digital signal processors. The rapid prototyping process is being developed for the ARPA/Tri-Services Rapid Prototyping of Application Specific Signal Processors (RASSP) program. The first application problem is to develop a virtual prototype for a real-time digital signal processor capable of forming images from high-resolution synthetic aperture radar data. Details of the application are provided along with design constraints and optimization requirements for the processor. The report also describes product and process metrics which are to be collected to derive measures of process and product performance. The application problem and associated performance metrics comprise what is termed a Benchmark Technical Description.

TABLE OF CONTENTS

TABLE OF CONTENTS (Continued)

LIST OF FIGURES

LIST OF TABLES

1. GENERAL

This document describes the technical requirements and deliverables for RASSP Benchmark-1. The main thrust of Benchmark-1 is the development of a virtual prototype for an embedded processor capable of forming radar images in real time for a high resolution synthetic aperture radar (SAR).

This section contains background information on the Advanced Detection Technology Sensor (ADTS), a Ka-band SAR sensor and data recording system operated by M.I.T. Lincoln Laboratory. Section 2 sets forth the requirements for a signal processor capable of forming SAR images in real-time from the ADTS sensor data. Section 3 describes executable requirements which compliment this document by providing VHDL behavioral models of the required processing and capturing the timing information for the external processor interfaces. Section 4 describes the metrics which must be collected to evaluate the performance of the RASSP process and products associated with the development of the virtual prototype. Deliverables are discussed in Section 5. The response of the Developers to this Benchmark Technical Description (BTD) is described in Section 6.

1.1 Introduction and Objectives

A component of the ARPA Rapid Prototyping of Application Specific Signal Processors (RASSP) program, is the execution by the RASSP Developers, Lockheed Sanders, Inc. and Martin Marietta Corporation, of application benchmarks. This document defines the first application benchmark, which is directed toward the development of a virtual prototype of an embedded processor for real-time SAR image formation.

1.1.1 Virtual Prototyping

The definition of the term "virtual prototype" is open to some interpretation. As used here, a virtual prototype is an executable software model of an embedded processor which represents all of the important function and timing information of the processor with sufficient fidelity to insure that the processor will perform as intended when constructed in accordance with the architecture underlying the virtual prototype model. In its full embodiment, a virtual prototype is a sufficiently trustworthy simulation or emulation of a real system that building the actual system simply to demonstrate feasibility is no longer necessary. However, a major deficiency of the virtual prototype relative to the corresponding physical prototype, is that the virtual prototype is generally incapable of sustaining real-time operation. To date, virtual prototyping with the level of fidelity implied in the preceding statements has generally been performed only at the chip level. The RASSP program seeks to extend the utility of the virtual prototype to the board and subsystem level.

One of the principal issues of interest in Benchmark-1 is the degree of fidelity and completeness that can be obtained for reasonable cost using existing virtual prototyping methodologies and models. The expectation is that IEEE-compliant VHDL simulation will be the vehicle for developing the virtual prototype. The level of detail and fidelity attained in the virtual prototype will be limited by the constraints of time (6 months) and level of effort (5000 person-hours) imposed for Benchmark-1. One of the responsibilities which the Developers have under the RASSP program is to establish cost-effective methodologies and tools for the creation and application of virtual prototypes to the development of embedded signal processing systems. For the purposes of Benchmark-1, the level of detail for function and timing incorporated in the virtual prototype should not extend beyond that of an instruction set architecture (ISA) model of programmable devices running application code written in a high-level source language such as Ada. The exception is that more detailed models may be required to validate interface and timing constraints. This level of detail will likely not be achievable within the time duration and level of effort constraints imposed on Benchmark-1. The ISA level of modeling represents a goal to progress toward, and defines the limit of detail expected in the VHDL modeling. It does not prohibit the Developer from incorporating more detailed models for reasons of availability or risk reduction. Prior to developing a detailed virtual prototype for a preferred architecture, less detailed modeling and evaluation of alternative architectures shall be performed to select the architecture which best meets the requirements and performance goals outlined in Section 2.

This document establishes the requirements for development and delivery of a virtual prototype of a real-time SAR image processor capable of interfacing to the ADTS sensor. The RASSP process will be applied to extract processor requirements and develop VHDL models for a preferred architecture within the benchmark cycle. Trade-off analyses will be performed for a minimum of two design architectures, one emphasizing low cost to prototype, and the other emphasizing lowest power and weight. It is anticipated that the low-cost design will minimize cost by utilizing commercial-off-the-shelf (COTS) products at the board level, while the low-power/weight design will reduce weight and power by employing chip-level COTS design. Inclusion of additional architectures in the initial trade-off analysis is encouraged. At least two architectures must be carried to the level of VHDL performance modeling to establish estimated performance.

While the development of hardware is not an aspect of Benchmark-1, the virtual prototype design should be developed with the view that hardware may be fabricated based on the virtual prototype in a subsequent six-month benchmark cycle. Therefore, the designs are constrained to be producable in unit quantities over a period of nominally six months, and for a total equivalent cost (normalized to person hours) of between 5000 to 10,000 person hours. This BTD includes size, weight, and power constraints consistent with use of the processor on board a small unmanned air vehicle (UAV); see Appendix D. The resulting processor will have a form factor consistent with a UAV, but will be compatible with the sensor flown on board the ADTS Gulfstream aircraft.

1.2 Overview of ADTS System

The ADTS system consists of an integrated radar, navigation, and recording system carried on board a Gulfstream twin-engine aircraft [1]. The ADTS aircraft and sensor are shown in Figure 1 and Figure 2, respectively. The ADTS system began operation in 1987 and has logged approximately 400 missions collecting data on a variety of terrain and target types. The database is currently archived and managed by Lincoln Laboratory, and selected segments of the data have been approved for public release. Images formed from the ADTS data have been made available to selected universities as part of work sponsored by ARPA on automatic target recognition algorithms.

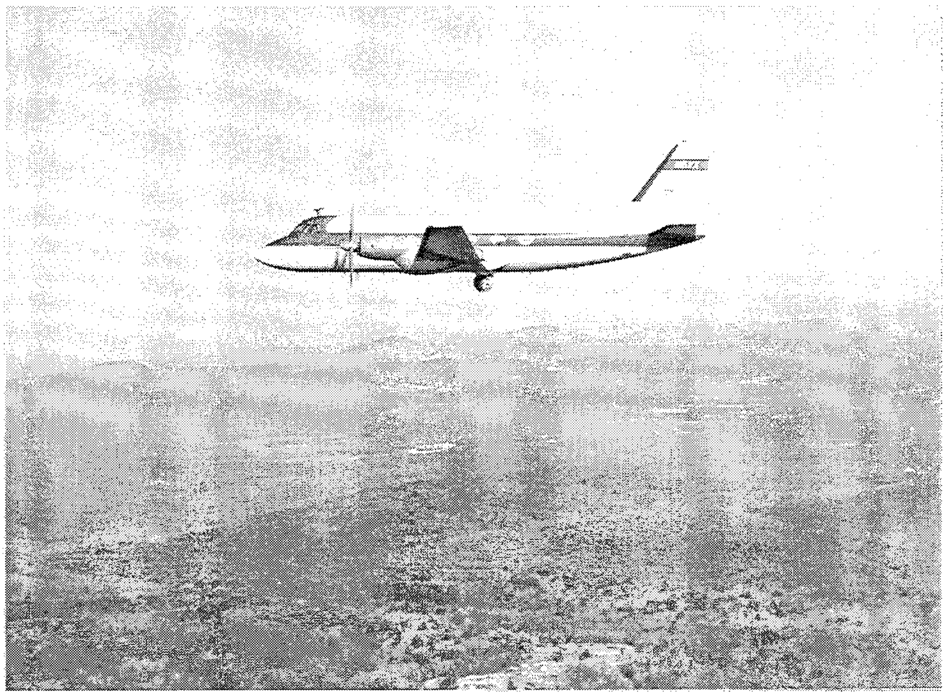


Figure 1. ADTS aircraft.

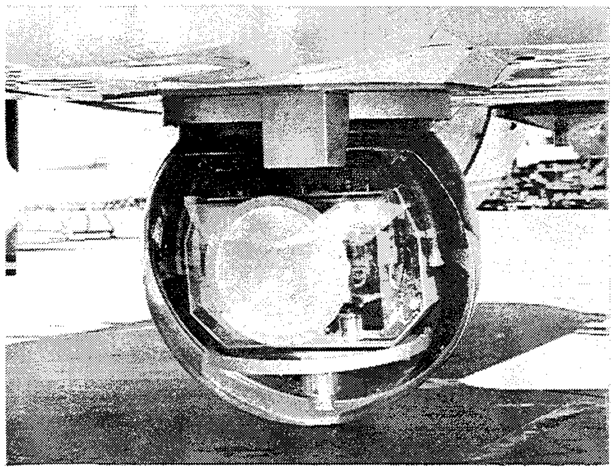


Figure 2. ADTS sensor.

To develop a processor for real-time SAR image formation from the ADTS sensor data, information about the radar characteristics and data formats is required. The intent is to interface the real-time SAR image processor directly to the ADTS system without modifying the existing data formats or timing of the ADTS system.

The ADTS radar is an air-to-ground synthetic aperture radar (SAR) operating in "stripmap" mode with a $\pm 90^o$ squint angle (i.e., the radar is pointed $\pm 90^o$ relative to the velocity vector). The radar has a center frequency of 33.56 GHz, a swath width of 375 m, and a nominal 7.26 km range to the center of the swath. The radar is fully polarimetric with interleaved H-pol and V-pol transmission and simultaneous H-pol and V-pol reception. The radar transmits at a 3 kHz rate, so that the same-polarization pulse repetition frequency (PRF) is 1.5 kHz as shown in Figure 3. The SAR resolution is 0.3 m.

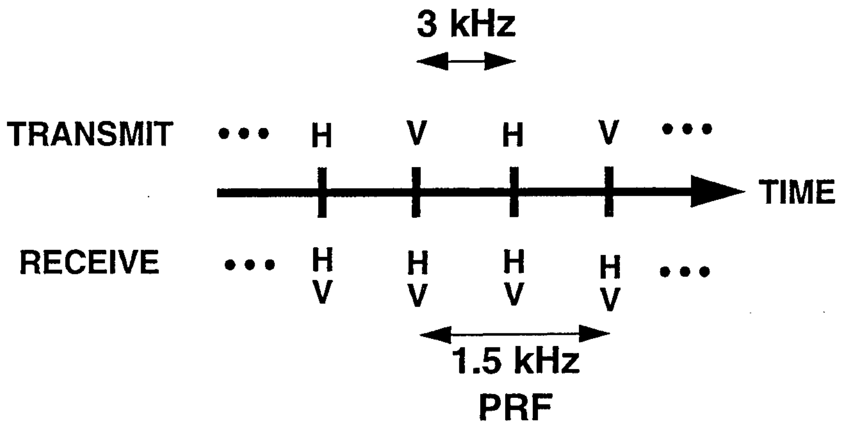


Figure 3. Interleaved horizontal and vertical transmit polarizations.

The radar uses stretch processing for wideband pulse processing. The linear frequency modulation (LFM) transmit pulse has a 600 MHz bandwidth and a 30μsec pulsewidth. Received pulses are de-ramped, down-converted, and filtered so that the inputs to the A/D converters are real, uncompressed (i.e., frequency domain) data at a video frequency of 31.25 MHz and a 50 MHz bandwidth. De-ramping is done using a 650 MHz, 32.5μsec LFM waveform, shown in Figure 4, which is longer than the transmit waveform by an amount which accommodates the 375 m range swath.

Received pulse data are sampled using 8-bit A/D converters at a 125 MHz rate, yielding 4064 real samples per pulse. The range window for the radar is 468.4 m, and the 375 m range swath is at the center of the range window.

After digitization, the pulse data are phase and frequency compensated for non-linear motion of the aircraft and timing errors in the de-ramp process. Moreover, pulses are Doppler processed and re-sampled

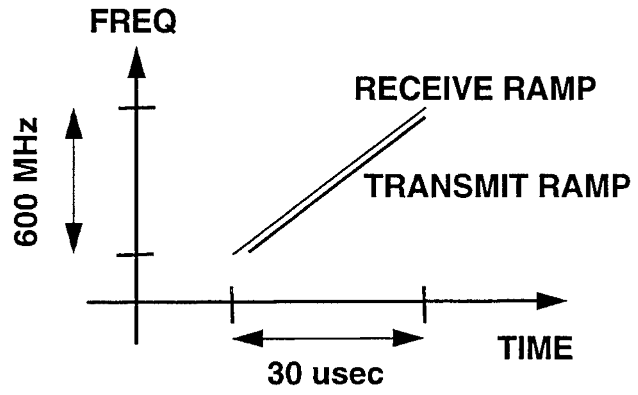


Figure 4. Transmit ramp and receive ramp for stretch processing.

(interpolated) to yield pulses at a constant spatial interval of 0.2287 m along the flight path. This represents a pulse PRF of 437 Hz for a nominal aircraft velocity of 100 m/s. The output of the re-sampling process are pulse data with 11 bits of precision. As described in Section 1.3, the re-sampled pulses are buffered and presented as a 40-bit wide word transmitted serially over a fiber optic link to the SAR processor.

1.3 Sensor Data Format

Figure 5 illustrates the organization of the data within the 40-bit data word as it is presented to the parallel-to-serial converter for transmission over the fiber-optic link. The 11-bit samples are right-justified and sign extended in a 12 bit field. There are 2032 even/odd data pairs comprising a pulse repetition interval (PRI), and four transmit-receive polarization pairs for each PRI.

1.3.1 PRI Preamble

Each PRI of data is preceded by a code or preamble which is duplicated in bits 3 through 16, 19 and 32 of the 40-bit data word. This preamble consists of a prefix of 5 leading zeros, followed by a 13-bit

39:33	32	31:20	19	18:17	16	15:04	03	02:00
	0	NOT USED	0		0	5 ZERO WORDS	0	
	B		B		B	13 BARKER WORDS	B	
	F		F		F	2 FILLER WORDS	F	
	H	**ODD HH SAMPLES** 2032 WORDS	H		H	**EVEN HH SAMPLES 2032 WORDS**	H	
	A		A		A		A	
	F	VARIABLE FILLER WORDS	F		F	VARIABLE FILLER WORDS	F	
	0	NOT USED	0		0	5 ZERO WORDS	0	
	B		B		B	13 BARKER WORDS	B	
	F		F		F	2 FILLER WORDS	F	
	H	**2032 ODD HV WORDS**	H		H	**2032 EVEN HV WORDS**	H	
	F	VARIABLE FILLER WORDS	F		F	VARIABLE FILLER WORDS	F	
	0	NOT USED	0		0	5 ZERO WORDS	0	
	B		B		B	13 BARKER WORDS	B	
	F		F		F	2 FILLER WORDS	F	
	H	**2032 ODD VH WORDS**	H		H	**2032 EVEN VH WORDS**	H	
	F	VARIABLE FILLER WORDS	F		F	VARIABLE FILLER WORDS	F	
	0	NOT USED	0		0	5 ZERO WORDS	0	
	B		B		B	13 BARKER WORDS	B	
	F		F		F	2 FILLER WORDS	F	
	H	**2032 ODD VV WORDS**	H		H	**2032 EVEN VV WORDS**	H	
	F	VARIABLE FILLER WORDS	F		F	VARIABLE FILLER WORDS	F	

0	ZERO BITS
B	BARKER BITS
H	HEADER BITS
A	AUX BITS
F	FILLER BITS

Figure 5. Format of the 40-bit wide ADTS data.

Barker code, followed by a suffix of 2 trailing don't-care words. The Barker code values, along with the zero prefix and don't-care suffix, are indicated in Table 1.

Table 1. Bit pattern for PRI preamble

Run Length	Bits 3-16, 19, 32
5	0
3	1
1	0
1	1
2	0
1	1
3	0
2	1
2	x

The sole function of the preamble is to indicate that a PRI of radar data follows. The use of a Barker code in the preamble does not relate in any way to any modulation applied to the radar waveform.

Bit 16 of the 40-bit input data word contains two types of information in bit-serial format. The information always starts with the HH PRI of the data, and fits entirely within the HH dataset.

- A 16 bit header word for each PRI identifying the transmit-receive polarization. The header word is recorded with the MSB as the first of 16 bits.
- Aux data consisting of 57 16-bit words with the MSB again recorded as the first bit for each word. The Aux data follows immediately after the header word.

This same bit-serial information is repeated in bit locations 3, 19, and 32 of the 40-bit input data word.

1.3.2 Bit-Serial Aux Data

Figure 6 defines the contents of the Aux record and the associated units. For the Aux variables that are written over two 16-bit words, the MSBs are contained in the first word.

1.3.3 Header Word

The header word is used to signify the transmit-receive polarization of the associated PRI of data. The four types of polarization and their associated header designations are given in Table 2.

	WORDS	LSB	DEFINITION	COMMENTS
PNINS	2	2^{-14} m	INS North Position	MSB of Word 2 is 0
PEINS	2	2^{-14} m	INS East Position	MSB of Word 2 is 0
PDINS	1	2 m	INS Down Position	
VNINS	1	2^{-5} m/s	Level North Velocity	
VEINS	1	2^{-5} m/s	Level East Velocity	
PNMS	2	2^{-14} m	Motion Sensed North Pos.	MSB of Word 2 is 0
PEMS	2	2^{-14} m	Motion Sensed East Pos.	MSB of Word 2 is 0
PDMS	2	2^{-14} m	Motion Sensed Down Pos.	MSB of Word 2 is 0
VNMS	2	2^{-22} m/s	Motion Sensed North Vel.	MSB of Word 2 is 0
VEMS	2	2^{-22} m/s	Motion Sensed East Vel.	MSB of Word 2 is 0
VDMS	2	2^{-22} m/s	Motion Sensed Down Vel.	MSB of Word 2 is 0
TRGN	2	2^{-14} m	Aimpoint North Pos.	MSB of Word 2 is 0
TRGE	2	2^{-14} m	Aimpoint East Pos.	MSB of Word 2 is 0
TRGD	2	2^{-14} m	Aimpoint Down Pos.	MSB of Word 2 is 0
TENSEC	1	2^{0} sec	Time Word #1	Time = 10 x TENSEC + MILSEC / 1000
MILSEC	1	2^{0} msec	Time Word #2	
SLTRNG	2	2^{-14} m	Range to Aimpoint	MSB of Word 2 is 0
RNGDOT	2	2^{-22} m/s	Velocity Toward Aimpoint	MSB of Word 2 is 0
ANTYAW	1	2^{-13} rad	Antenna Yaw Command	
ANTPIT	1	2^{-13} rad	Antenna Pitch Command	
ANTROL	1	2^{-13} rad	Antenna Roll Command	
ANTSTA	1		Antenna Status Flag	
SCNPOS	1	2^{0} steps	Scanner Position	8192 steps = 360°
RAW 7	1	2^{1}	Range to DME7	
N1	1	2^{1}		
N2	1	2^{1}		
CNAVER	2	2^{-14} m	Avg. North Update for INS	MSB of Word 2 is 0
CEAVER	2	2^{-14} m	Avg. East Update for INS	MSB of Word 2 is 0
WMC	1	2^{9} Hz	MoComp Freq. Coef.	
PHSMC	1	2^{-16} cycles	MoComp Phase Coef.	
RAI	1	2^{0}	Azm. Prefilter Coef.	
HDGINS	1	$2^{-15}\pi$ rads	Heading Angle	
PCHINS	1	$2^{-15}\pi$ rads	Pitch Angle	
ROLINS	1	$2^{-15}\pi$ rads	Roll Angle	
MODE	1		Radar Mode	
CMPTME	1	2^{0} ms	Time in msec	
IMUVX	1	2^{-14}	IMU x velocity	units of .3048 m/s
IMUVY	1	2^{-14}	IMU y velocity	units of .3048 m/s
IMUVZ	1	2^{-14}	IMU z velocity	units of .3048 m/s
IMUTHX	1	2^{-19} rad/s	IMU Neg. Head. Ang. Rate	
IMUTHY	1	2^{-20} rad/s	IMU Neg. Roll Ang. Rate	
IMUTHZ	1	2^{-21} rad/s	IMU Neg. Pitch Ang. Rate	

Figure 6. Aux record format and units.

Table 2. Polarization and hexadecimal header designations.

Polarization	**Designation (Hex)**
HH	03x0
HV	43x5
VH	83xA
VV	C3xF

x - don't care

2. PROCESSOR REQUIREMENTS

This section describes the complete requirements for a real-time, multiple polarization SAR processor. For Benchmark 1, processor development will only be carried to the point of a virtual prototype that utilizes VHDL models running in non-real time.

2.1 Image Formation

During operation, the SAR processor continuously forms images for up to three of the four input polarizations (Figure 5). Figure 7 illustrates the processing flow. In addition to the continuous process of image formation, there is a setup process which occurs before the first frame of data. The setup process is described in more detail in Section 2.2.2.

2.1.1 Accuracy

The SAR processing hardware must preserve the signal-to-noise ratio (SNR) inherent in the image data, and must not introduce appreciable computational noise or artifacts into the image. The SAR processor should be capable of supporting full-scale input signals without saturation, and quantization errors in the output data should be 10 dB below receiver noise.

A full-scale input signal has an amplitude of approximately 2^{10} relative to the input LSB of 2^0. Receiver noise at the input is nominally set at the 5th bit, so that the noise standard deviation is $\sigma_n = 2^4$. This results in an input signal-to-noise ratio (SNR) of 2^{11}, or 33 dB. The peak pixel power for a full-scale input signal is 1.4736×10^{18} while the receiver noise power is 7.466×10^8. As a result, the peak output SNR is 1.979×10^9, or 93 dB, and the minimum dynamic range of the SAR processor is required to be 103 dB. Lincoln Laboratory has a non real-time implementation of the image formation algorithm which executes on a workstation. The images formed using this implementation are output in 32-bit IEEE floating point and represent the basis for acceptance testing of the SAR processor.

To verify that adequate processor dynamic range and accuracy are achieved, differences between corresponding pixels in a processed SAR image, x_{SAR}, and the Lincoln Laboratory reference SAR image, x_{REF}, will be calculated. If x_{SAR} and x_{REF} are complex pixel values from corresponding processed and reference SAR images, the power of the error due to processing, P_{ERR}, is given by

$$P_{ERR} = \frac{|x_{SAR} - x_{REF}|^2}{2}, \tag{1}$$

where the processing error in both the processed and reference imagery are equal. A processor dynamic range of 103 dB means that the P_{ERR} should be less than -103 dB relative to the maximum output signal power, P_{MAX}, or

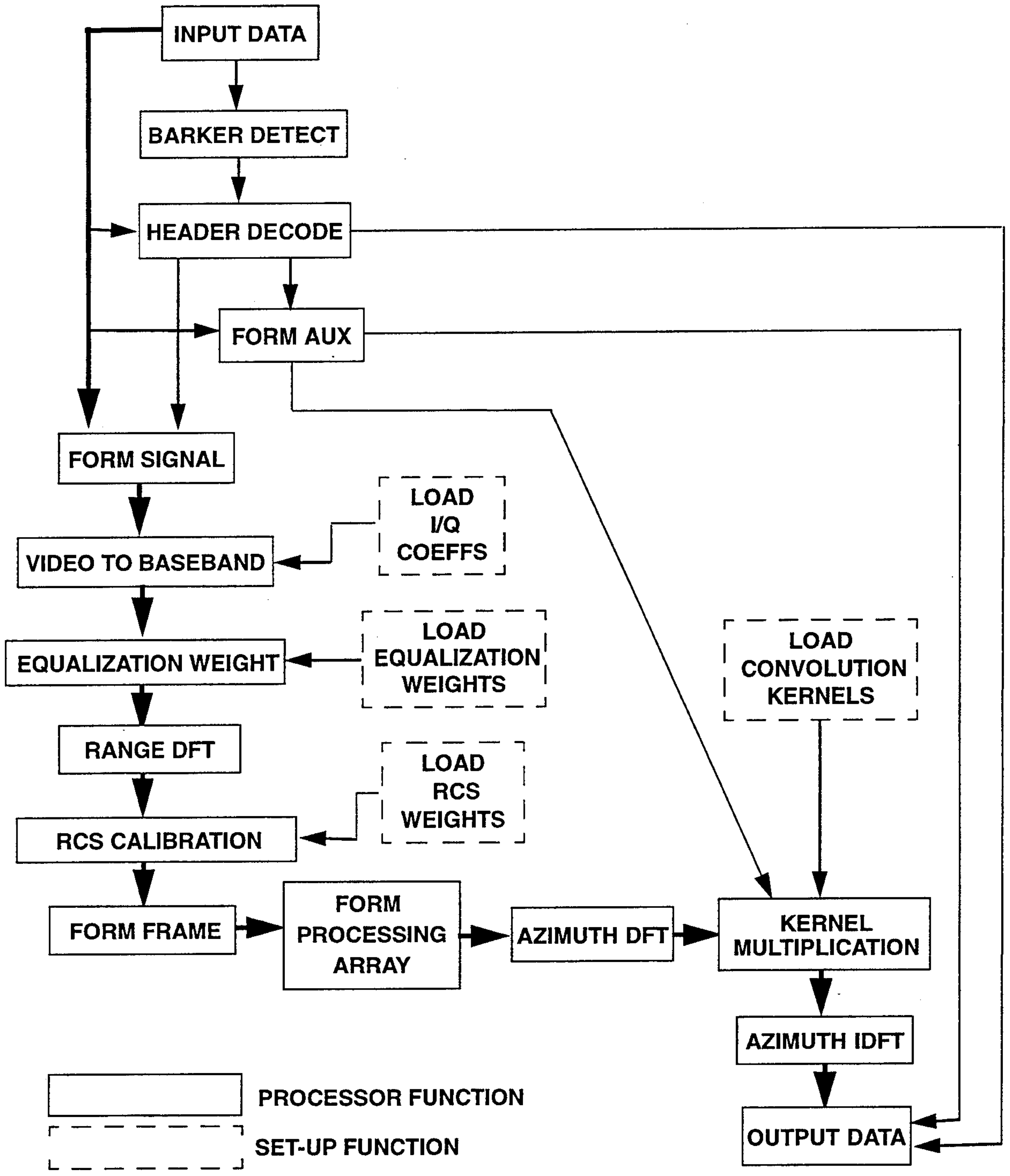


Figure 7. SAR image processing flow.

$$10\log\left(\frac{P_{ERR}}{P_{MAX}}\right) = 10\log\left(\frac{|x_{SAR} - x_{REF}|^2}{2P_{MAX}}\right) \leq -103, \qquad (2)$$

where P_{MAX} is the maximum pixel power of 1.4736×10^{18}. The SAR processor will be tested using both actual radar data and synthesized test data, with representative data sets supplied by Lincoln Laboratory in the tape media format discussed in Section 2.6.2.

2.1.2 PRI Detection

As previously discussed, channels of polarized pulse data, header data, and Aux data are presented to the RASSP processor. A 20-word sequence consisting of a 13-word Barker code with 5 leading zeros and 2 trailing don't-care words indicates the start of pulse data. This preamble is followed by 2032 words of 11-bit even pulse samples and 11-bit odd pulse samples, sign extended to 12 bits. Included with the pulse samples are header data and Aux data recorded in bit-serial fashion and duplicated in bit positions 3,16,19, and 32 of the 40-bit data word. Header data describes the polarization of the pulse data, and Aux data contains ancillary navigation and radar data. Pulse data for the four polarizations are output in a repeated sequence (i.e., ... , HH, HV, VH, VV, HH, ...), but Aux data are only written for the leading pulse of the sequence (i.e., HH). There are filler data between the end of data for one pulse and the start sequence for the next pulse.

PRI data are written at a 4.56 MW/s rate. The maximum PRF is 556 Hz which corresponds to an aircraft ground speed of 127.1 m/s. The maximum PRF is needed if strong tail-winds are present during the flight. Similarly, the minimum PRF is 200 Hz and corresponds to an aircraft ground speed of 45.7 m/s.

Because the HV and VH polarizations contain the same information, no more than three of the four polarizations are used to form images. Generally, images will be formed for the HH and VV polarizations, and either the HV or the VH polarization as established through the RS-232 control interface by the operator. It is therefore necessary to process and retain pulse polarization information from the header. In addition, Aux data must be processed to extract slant range data for each pulse (SLTRNG) as well as antenna squint information.

Squint angle is the difference between the direction of the line-of-sight from the radar to the ground and the heading of the aircraft. Aircraft heading, in units of degrees, is calculated from the sensed[1] aircraft velocity vector by

$$\text{Heading} = 90 - \tan^{-1}\left(\frac{\text{VNMS}}{\text{VEMS}}\right) \qquad (3)$$

1. Sensed aircraft position and velocity vectors are derived from on-board IMU, INS, and GPS data and represent best estimates of aircraft position and velocity.

where VNMS and VEMS are components of the aircraft velocity vector in the North and East direction, respectively. VNMS and VEMS are variables in the Aux data record. The direction of the line-of-sight is derived from the vector to the aimpoint and the aircraft position vector. Direction of the line-of-sight, in units of degrees, is calculated from sensed aircraft position and aimpoint by

$$\text{Line-of-sight} = 90 - \tan^{-1}\left(\frac{\text{TRGN}-\text{PNMS}}{\text{TRGE}-\text{PEMS}}\right) \tag{4}$$

where PNMS and PEMS are components of the aircraft position vector in the North and East directions, respectively. Similarly, TRGN and TRGE are components of the aimpoint vector in the North and East directions. PNMS, PEMS, TRGN, and TRGE are variables in the Aux data record. Squint angles of $+90^o$ and -90^o indicate the antenna is pointed out the right or left side of the aircraft, respectively.

2.1.3 Video to Baseband I/Q Conversion

Prior to pulse compression, 4064 real video samples of each of the three polarizations to be imaged must be converted to complex (in-phase and quadrature, or I/Q) data at baseband. The baseline approach for performing the I/Q demodulation is to form sequences of even and odd pulse samples and modulate each sequence by $(-1)^n$. This yields two real-valued sequences for each pulse: an even sample sequence, $\{s_0, -s_2, s_4, -s_6, \ldots, -s_{4062}\}$, and an odd sample sequence $\{s_1, -s_3, s_5, -s_7, \ldots, -s_{4063}\}$, where s_n is the nth real sample. Currently, the even sequence is passed through an 8-coefficient FIR filter to yield the sequence $\{s_{i0}, s_{i1}, s_{i2}, \ldots, s_{i2023}\}$. The FIR filter output sequence is 8 samples shorter than the FIR input sequence because the filter must be initialized before valid data samples are obtained. Similarly, the odd sequence is passed through an 8-coefficient FIR filter to yield the sequence $\{s_{q0}, s_{q1}, s_{q2}, \ldots, s_{q2023}\}$. These sequences are combined to form the sequence of complex samples $\{(s_{i0},s_{q0}), (s_{i1}, s_{q1}), \ldots, (s_{i2023}, s_{q2023})\}$. The baseline coefficients for the FIR filters are given in Table 3. The current implementation of the FIR filter is given by,

$$y_n = \sum_{m=0}^{7} a_m x_{n+m} \tag{5}$$

where y_n is the nth output sample, x_n is the nth input sample, and a_m is the mth FIR coefficient.

2.1.4 Range Compression

In pulse compression, the 2024 (uncompressed) I/Q samples of each pulse are transformed into a (compressed) range pulse with 2048 samples. Each of the 2048 samples can be thought of as constituting a range-gate. The first step in pulse compression is the application of weighting to the amplitude of the complex valued I/Q data. This weighting reduces the sidelobes of the compressed pulse and is applied to the 2024 complex input samples with trailing zero-pads to expand the data to 2048 samples. Currently, Taylor weights are used for sidelobe reduction (see Appendix B). The complex input data are modulated by $(-1)^n$

Table 3. Baseline I/Q filter coefficients.

Index	Even Sequence	Odd Sequence
0	–0.021133	0.019827
1	0.055895	–0.011912
2	–0.148449	–0.067483
3	0.406139	0.917516
4	0.917516	0.406139
5	–0.067483	–0.148449
6	–0.011912	0.055895
7	0.019827	–0.021133

where n = 0, ... , 2047, so that the compressed range pulses are centered in the range window. This modulation is incorporated into the weights.

Prior to pulse compression, it is necessary to compensate non-ideal IF filter characteristics that degrade image resolution. This equalization is incorporated into the weights, and the combined complex equalization weights are down-loaded to the SAR processor prior to real-time operation via the control interface. In addition, these weights are polarization specific so that polarization data extracted from the data header must be used in establishing which set of equalization weights to apply to a given set of data.

Weighted I/Q data are transformed to (compressed) range data using a 2048 point DFT. The resulting sample interval in range is 0.2287 meters. In some cases, it may be necessary to compensate for elevation beam-shape modulation of the radar cross section (RCS) across the range swath. In addition, R^4 losses can produce as much as a 1 dB variation in RCS over the range swath. To compensate for these RCS variations, amplitude weights are applied to samples of the compressed pulse. The RCS weights are down-loaded to the SAR processor prior to real-time operation via the control interface.

2.1.5 Azimuth Compression

Azimuth compression is performed using the process of cross-range convolution filtering as shown in Figure 8. Compressed pulses are placed, in time sequence, into a 2-D processing array and each row of the array is convolved with a row-specific reference kernel. The convolution outputs are saved in an image array which becomes the output stripmap image of the SAR.

In this process, compressed pulses are placed in time sequence in a 2-D array referred to as a *frame*. Each row of the frame contains 512 pulses and each column contains 2048 range gates, so that each frame is a 2048×512 array. Once 512 pulses have been accumulated to form a frame, the frame is shifted into

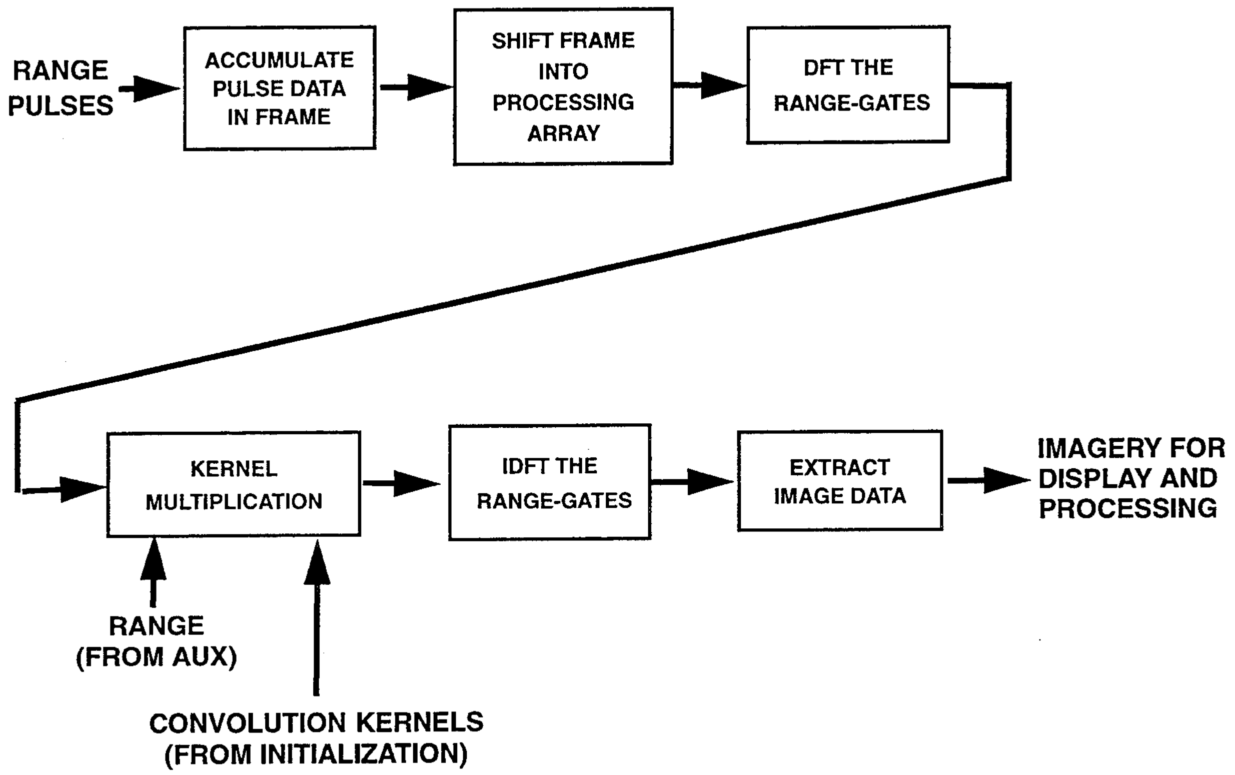

RANGE PULSES
ACCUMULATE PULSE DATA IN FRAME
SHIFT FRAME INTO PROCESSING ARRAY
DFT THE RANGE-GATES
KERNEL MULTIPLICATION
IDFT THE RANGE-GATES
EXTRACT IMAGE DATA
IMAGERY FOR DISPLAY AND PROCESSING
RANGE (FROM AUX)
CONVOLUTION KERNELS (FROM INITIALIZATION)

Figure 8. Azimuth compression processing.

the processing array. The *processing array* is a 2-D array where azimuth compression processing is performed. The processing array consists of two frames; i.e., the processing array is 2048×1024. As each new frame is shifted into the processing array, the oldest frame is shifted out. A convolution is performed along each row of the processing array, where the convolution kernel is the approximate response of a point scatterer located at the range-gate of the row, as described in Appendix A.

Figure 9 depicts convolution processing for one row of the processing array. Convolution processing is currently performed using DFTs with the overlap-save method. The processing array consists of 1024 pulses, with 2048 complex range samples each. The convolution kernels (see Appendix A) are 512 points long, but trailing zeros are used to pad-out the kernel to 1024 points. A 1024-point DFT of each row is multiplied with the 1024-point DFT of the associated convolution kernel. Each vector product is inverse transformed, and the last 512 samples of each inverse transform represent valid convolution outputs and are saved in an image array. A new frame is then shifted into the processing array, the convolution process is repeated, and more data are added to the image array thereby generating the output stripmap SAR image.

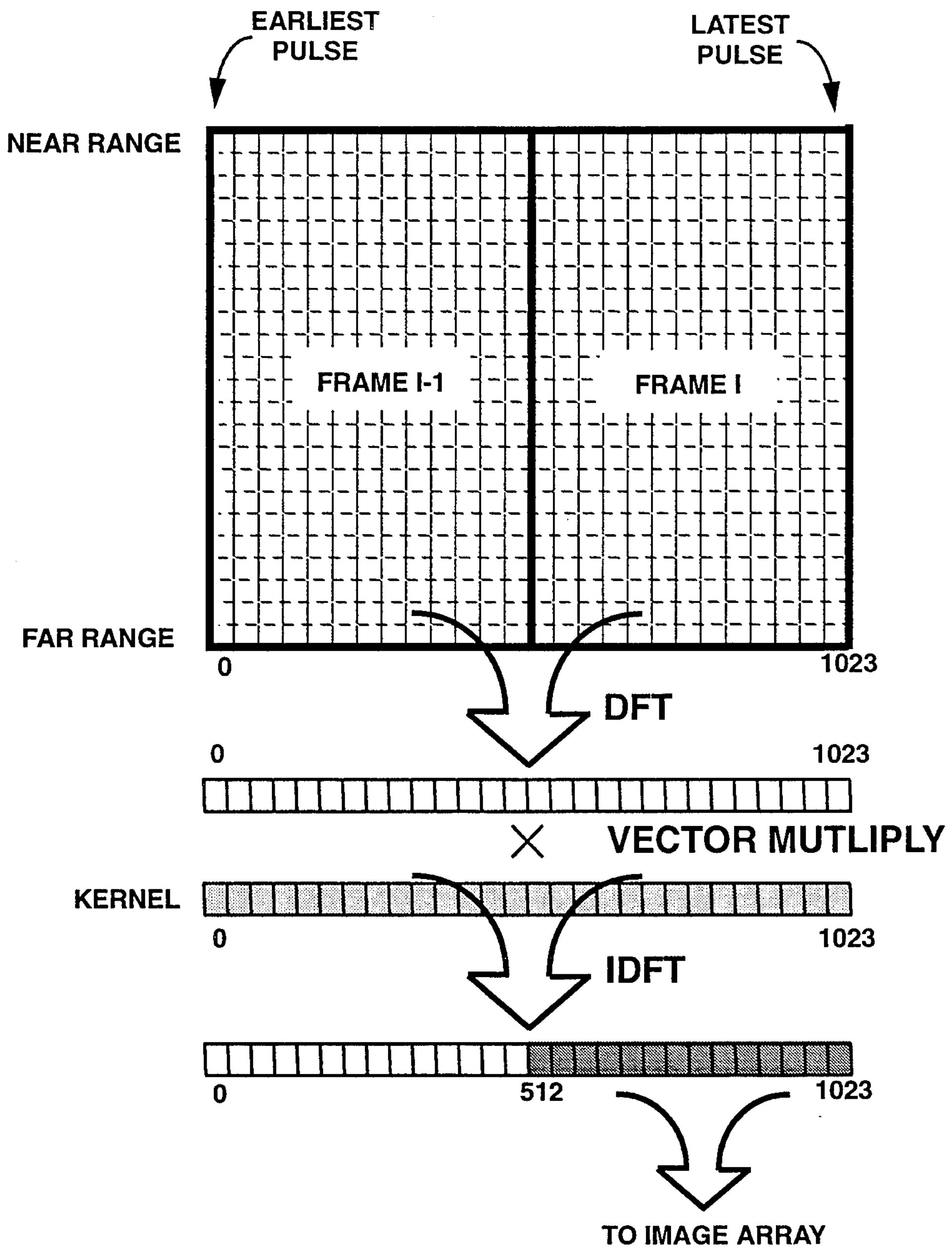


Figure 9. Cross-range convolution processing

The convolution kernel used for each row is selected from a database of 31 pre-calculated kernels, where the kernels have been Taylor weighted, zero padded, and Fourier transformed (i.e., the processor does not transform the kernels). The choice of kernel is determined by the slant range to the middle of the most recent frame in the processing array. This slant range is given by SLTRNG in the Aux record of the 256th pulse of the most recent frame. The kernels are calculated based on the slant range (SLTRNG) to the middle of the first frame of data obtained for a given pass, and are stored for use throughout the pass. Only 16 of the 31 stored kernels are used at one time in azimuth compression, but these 16 vary with each frame. A more detailed discussion of the kernel calculation and selection process is given in Appendix A. A typical SAR image output is shown in Figure 10.

2.1.6 Latency

The latency between a frame of data being input to the SAR processor and the corresponding image being output from the processor shall not exceed 3 seconds. A frame is considered to be input to the processor when the last pulse used to form the frame is passed to the SAR processor. A frame is output when the first image pixel from the corresponding frame is passed out of the processor. Figure 9 depicts the relationship between input frames of data and corresponding output frames of images.

2.2 External Interfaces

2.2.1 Fiber Optic Interfaces

The data stream to and from the SAR processor will be bit serial over fiber-optic links, compatible with TriQuint's HRC-500FS module. A data sheet describing the HRC-500FS, which is based on the HodRod^{TM} chip set, is given in Appendix C. Use of the TriQuint HotRod chip or transmit-receive module, in the signal processor is recommended, but any implementation of the interface compatible with the HRC-500FS, configured as described in this section, is acceptable. The description in the remainder of this section will assume that an HRC-500FS is used. To facilitate loop-back testing, the same data rate settings will be used for input and output data transfers. It will be shown subsequently that the HRC-500FS provides a link with excess capacity.

The data rate for the HRC-500FS will be derived from a reference clock of 25 MHz. DIV1, DIV0 will be set to 1, 0 respectively—corresponding to a data rate of 12.5 Mwords/s, 500 Mbps and 625 Mbaud (the link uses 4B/5B encoding). When the HRC-500FS is operated at 90% of maximum capacity or less, a simplified interface design with free-running data strobe is possible. This approach will be used for the SAR application, and provides an 11.25 MW/sec transfer rate capability which is suitably in excess of system requirements.

The transmission medium is a heavy-duty multi-mode 50/125 fiber optic cable with FSMA connectors. This cable runs to a bulkhead feedthrough (e.g., AMP #504020-2) on the SAR processor chassis, then through an adapter cable (e.g., Powell Electronics #907-99999-00264) to the HRC-500FS.

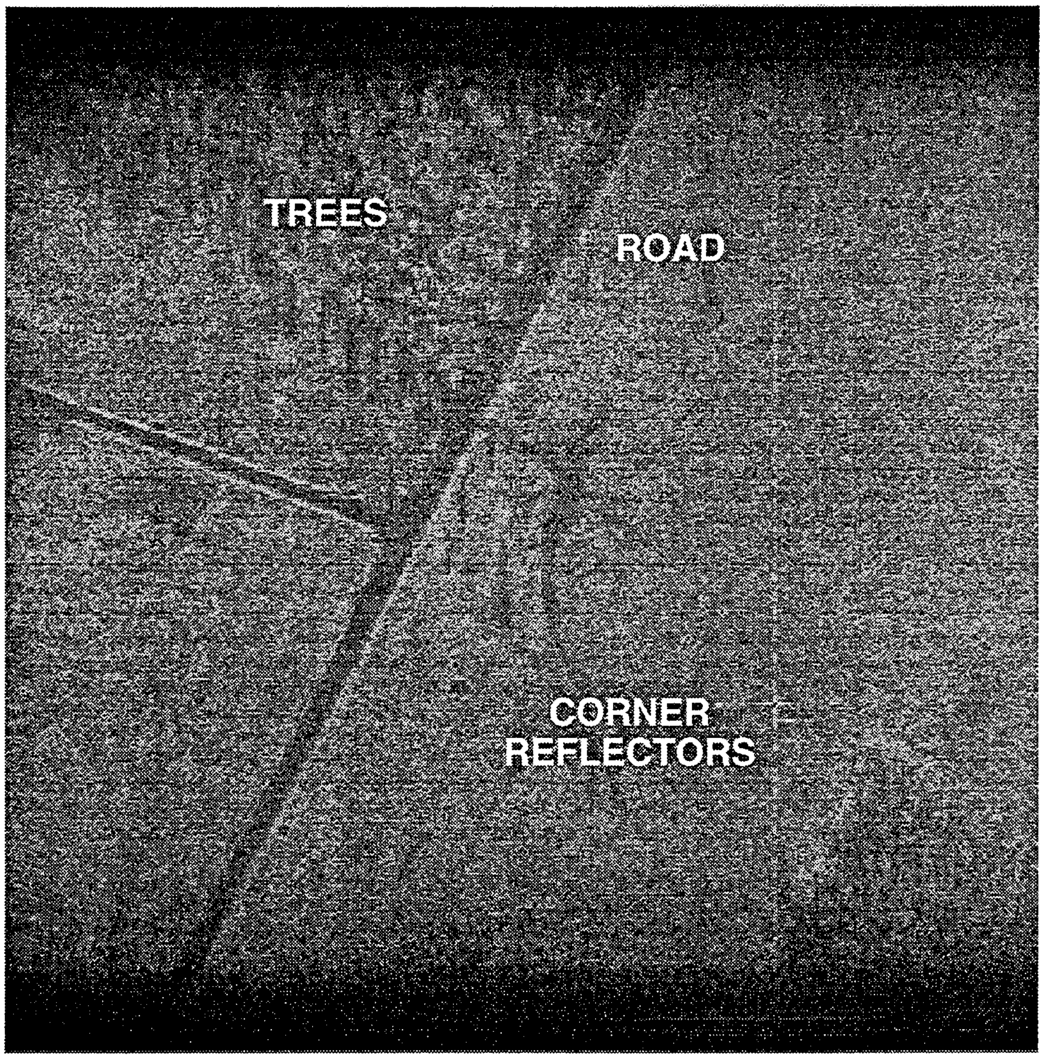


Figure 10. Typical SAR image.

2.2.1.1 Sensor Input Data. Referring to Figure 5, the input data fields of interest and their corresponding bit assignments on the HRC-500FS are summarized in Table 4. Bits 0-2, 17-18 and 33-39 are presently unused and will always be zero. As described in Section 2.1.2, data are available to the signal processor as 40-bit words at a rate of 4.56 MW/s, which is well within the limitations imposed by the

HRC-500FS.

There will also be extra (unused and meaningless) data words at the end of each PRI. The total number of these extra words will depend on the speed of the platform, which affects the spatial sampling rate or PRF. The PRI preamble should always be used to identify the beginning of a block, and a count of the number of input samples should be used to determine when the end of a PRI data block has been reached. A PRI will always consist of 2032 valid 40-bit data words, followed by a variable number of meaningless words.

Table 4. Input Bit/Pin Assignments.

Bits	Pins	Description
02 : 00	RxD02 : RxD00	Not used, always zero
03	RxD03	Aux Serial
15 : 04	RxD15 : RxD04	Even Sample
16	RxD16	Aux Serial
18 : 17	RxD18 : RxD17	Not used, always zero
19	RxD19	Aux Serial
31 : 20	RxD31 : RxD20	Odd Sample
32	RxD32	Aux Serial
39 : 33	RxD39 : RxD33	Not used, always zero

2.2.1.2 Processed Output Data. Processed image data will be output in the full precision of the SAR processor hardware, up to, but not exceeding, 32-bit floating point. If the chosen format uses less than 32 bits, the bits used will be justified into the LSBs so that the unused bits are the MSBs. IEEE single-precision floating point format is preferred.

The output data format for each polarization processed will consist of an image frame header followed by the complex samples from each image. There will be a maximum of three polarizations imaged per frame. The complex samples, corresponding to image pixels, will be output in azimuth order (i.e., in order of increasing time) beginning at the near range and finishing at the far range. The output data format is shown in Figure 11. The frame header will be constructed as shown in Figure 11 with the format as shown in Table 5.

39:38	37:32	31:16	15:00
2-BIT POLARIZATION		5 WORDS OF LEADING ZEROS	
		13 WORD BARKER CODE	
		2 WORDS OF FILLER	
		POLARIZATION CODE	REPEATED POL CODE
		AUX WORD 1	REPEATED AUX WORD 1
		⋮	⋮
		AUX WORD 57	REPEATED AUX WORD 57
		I PIXEL VALUE; RANGE 0, AZIMUTH 0	
		Q PIXEL VALUE; RANGE 0, AZIMUTH 0	
		⋮	
		I PIXEL VALUE; RANGE 0, AZIMUTH 511	
		Q PIXEL VALUE; RANGE 0, AZIMUTH 511	
		I PIXEL VALUE; RANGE 1, AZIMUTH 0	
		Q PIXEL VALUE; RANGE 1, AZIMUTH 0	
		⋮	
		I PIXEL VALUE; RANGE 2047, AZIMUTH 511	
		Q PIXEL VALUE; RANGE 2047, AZIMUTH 511	

Figure 11. Format of the 40-bit wide output data.

The zero prefix and Barker code will be implemented across the 32 LSBs of the parallel output word. The polarization code for the image will be specified according to the code in Table 2. Each two-byte code defining the polarization will be embedded in the 16 LSBs of a 40-bit output word, and repeated in the 16 contiguous bits. The 57 Aux data words output with each frame will be those associated with the 256th pulse of the most recent frame in the processing array, as discussed in Section 2.1.5. Each two-byte Aux data word will be embedded in the 16 LSBs of a 40-bit output word and repeated. Each complex image sample will be output as a real sample embedded in a 32-bit output word followed by an imaginary sample

Table 5. Format of the 40-bit wide output data.

Word No.	Bits	Pins	Contents
1 - 5	31 : 00	TxD31 : TxD00	Prefix of Zeros
6 - 18	31 : 00	TxD31 : TxD00	Barker Code
19 - 20	31 : 00	TxD31 : TxD00	Suffix (don't care)
21	15 : 00	TxD15 : TxD00	Polarization Code
21	31 : 16	TxD31 : TxD16	Polarization Code
22 - 78	15 : 00	TxD15 : TxD00	Aux Data
22 - 78	31 : 16	TxD31 : TxD16	Aux Data
79 - 2,097,230	31 : 00	TxD31 : TxD00	Complex Image containing 512 × 2048 pixels
1 - 2,097,230	39 : 38	TxD39 : TxD38	2-bit polarization code: 0→HH, 1→HV, 2→VH, 3→VV

embedded in a 32-bit output word. The LSB of the image samples is bit 00. The total number of bits in an output word required to represent a real or imaginary sample will depend upon the number representation used in the SAR processor hardware, but must not exceed 32. If the output words are less than 32 bits they should be right justified in the bit 31:00 field (the MSBs will be the unused ones).

The output bits 39:40 are used for a 2-bit polarization code, as shown in Table 5. This 2-bit code is in addition to the 16-bit code in bits 31:16 and 15:00 of output frame word 21. This 2-bit code is present with every output word. It is expected that this will be implemented by having a 2-bit output register. The data can be output in bursts with the contents of the 2-bit register being changed between bursts.

At the maximum PRF of 556 Hz, the 512 pulses needed to form an image frame are collected in slightly more than .92 s. If three images, corresponding to three different polarizations, are output at this same rate, then the output interface must support an average transfer rate of slightly more than 6.83 MW/s or equivalently 27.32 MB/s (34.15 MB/s including the unused byte of the 40-bit word), which represents the maximum average output rate. Thus, the input rate (4.56 MW/s) and output rate (6.83 MW/s) are well within the 11.25 MW/s link capacity. The output data may be output in bursts at any rate between 6.83 and 11.25 MW/s.

2.2.2 Control and Diagnostic

The control interface is bidirectional RS-232. The baud rate will preselectable. At a minium the choices will include 9.6 Kbaud and 19.2 Kbaud. It is desirable that the interface also support 38.4 Kbaud. Hardware handshaking will be used, and the interface will support the transfer of 8-bit binary data—although initially we do not plan to transfer binary data.

The RS-232 connection on the SAR processor will be configured in the same fashion as data communications equipment. Consequently, it will likely be possible to connect directly (with no pin swaps) to the modem output of a workstation or personal computer. The connector will be male, and the pin assignments will be made as indicated in Table 6.

Table 6. Control and diagnostic pin assignments

Direction*	Pin	Signal	Function
B	1	FG	Frame ground
T	2	TD	Data to processor
F	3	RD	Data from processor
T	4	RTS	Flow control - OK for processor to send
F	5	CTS	Flow control - processor is ready for data
F	6	DSR	Always true
B	7	SG	Signal ground
F	8	DCD	Always true
T	20	DTR	Not used
* Direction denotes the data flow to and from the processor; T - into, F - from, B - bidirectional.			

The control and diagnostic interface is intended to serve a variety of purposes. The minimum set of control and diagnostic commands which must be supported are listed in Table 7. Additional commands may be added at the discretion of the developer as an aid in diagnosing and debugging the system during development.

The interface is intended to allow a person to sit at a terminal and enter the commands, although it may not be practical for a person to type in the data sets required by some commands. Also, an external computer should be able to "drive" the SAR processor using this same interface. To this end, all of the commands in Table 7 and data for those commands will be in ASCII. Commands will be entered as the strings given in Table 7 terminated by a CR (carriage return). The commands will not be case sensitive. Backspace characters (control-H) will delete the most recent characters entered. Each line will be terminated by a CR. For those commands that require data, an EOF (end of file, indicated by a control-D) after the CR for the last line will indicate the end of the numerical input for a command. If an incorrect number of values is input, an error message will result.

Table 7. Control and diagnostic interface commands.

Command	Number of data lines	Type	Function
Reboot	—	—	Reboot the processor
Restart	—	—	Restart the processor
Init	2	Integer	Load various control registers
Run	—	—	Form images on a continuous basis
Stop	—	—	Stop execution and wait for command
Status	—	—	Dump system status message
Step	—	—	Process one image frame and stop
StepN	1	Integer	Process N image frames and stop
Debugger	—	—	Enter debugger
Loadref	31,744	Complex	Load reference kernels
Loadequal	8,192	Complex	Load Equalization weights
Loadiqeven	48	Real	Load I/Q filter weights for even sequence
Loadiqodd	48	Real	Load I/Q filter weights for odd sequence
Loadrcs	2048	Real	Load RCS weights
Dumpinit	2	Integer	Dump parameters input by Init command
Dumpref	31,744	Complex	Dump reference kernels
Dumpequal	8,096	Complex	Dump Equalization weights
Dumpiqeven	48	Real	Dump I/Q filter weights for even sequence
Dumpiqodd	48	Real	Dump I/Q filter weights for odd sequence
Dumprcs	2048	Real	Dump RCS weights
Selftest	—	—	Run Self test once
SelftestN	1	Integer	Run multiple passes of selftest
Linktest	—	—	Run test of fiber interface once, with no external fiber loopback cable in place
LinktestN	1	Integer	Run multiple passes "linktest"
Linktestf	—	—	Run tester of fiber interface once, using a loopback cable
LinktestfN	1	Integer	Run multiple passes of fiber interface test

The types of numerical input for each command are given in Table 7. Integers will be entered as decimal integers. Real numbers can be entered either with or without an exponent being specified. Here are some examples of real numbers: "1234.012", "-0.987895", ".1239876", "1.334455e+01". Complex numbers will be entered as two real numbers on the same line, separated by a space; with the real part entered first.

When the processor is ready for another command it will respond with a prompt. The prompt will be "SAR>". If an illegal command is entered the processor will respond with an error message, preceding the prompt. All error messages will start with a "?" as the first character on the line.

During acceptance testing, it is likely that the SAR processor will be controlled using the same workstation that controls the data source and sink. Here is a representative sequence of events for testing the SAR processor:

1. The workstation issues the "reboot" command. This initializes the processor.
2. The workstation issues a "stop" command. As discussed in Section 2.2.2.1, the "reboot" command puts the processor in a mode where it is ready to accept data. One reason for issuing a "stop" command is so that a "run" command can be given later. The prompt back from this later "run" command will indicate that the SAR processor is ready to accept data.
3. The workstation issues the "Init" command.
4. The workstation issues the commands "loadref", "loadequal", "loadiqeven", "loadiqodd", "loadrcs". This loads the required constants into the processor. This step involves the transfer of on the order of a million characters. At 9.6 Kbaud this is likely to take approximately 17 minutes or around 4 minutes at 38.4 Kbaud—assuming that the workstation and processor are able to exchange data at full speed.
5. When it is necessary to check the setup data to the SAR processor, the workstation issues dump commands to read the data back.
6. The workstation issues a "run" command. The SAR processor indicates that it is ready for data by giving the specified prompt.
7. The workstation starts the data sink. This is the interface and disk subsystem used to store data from the SAR processor.
8. The workstation starts the data source. This is the interface and disk subsystem used to send data to the SAR processor.
9. The data source stops after the test data has been sent to the processor.
10. The workstation issues a stop command to the processor. The processor indicates it is finished processing data by giving a prompt to this command. It is not necessary for the stop command to verify all data are processed—i.e. it would be acceptable to leave the processing of the last few frames incomplete if it reduces the complexity of the processor. There will be a few frames of source data whose images will not be checked during acceptance testing.
11. Once the workstation receives a prompt from the stop command, it shuts down the data sink.

12. The workstation compares processor image data from the data sink with reference image data. Only the first N frames of data will be compared, where N is no more than three frames less than the number of frames sent by the source.

2.2.2.1 Reboot. This command is intended to be the same as a power up initialization. The processor should do the following:

1. Run the self test diagnostics—this is the same as the command “selftest”.
2. Flush all buffers.
3. Initialize the setup constants (items initialized by the “init”, “loadref”, “loadequal”, “loadiqeven”, “loadiqodd”, “loadrcs”) to reasonable values. These initial values may be “hardwired” into the program so that the processor may generate images without setup constants being loaded from an external device. The resulting images may have poorer quality than they would if a set of setup constants were loaded, via the control interface, that correspond to the specific data set being processed. Alternatively, the setup constants could be loaded from a non-volatile memory. The constants would be loaded into non-volatile memory by a command which copies the current setup constants into memory.
4. Start accepting input data if there is any—this is the same as a “run” command.

The SAR processor must be able to “come-up” (execute a reboot command) while data is streaming into its input port. The processor may ignore this data until it is up and ready to go. The processor will indicate “ready” by giving a prompt on the control interface. If data is already streaming into it at this time, the processor will pick up at the start of the next PRI. Note that if data is streaming into the processor while it is coming-up, it will not be well defined as to where in the stream of incoming data the processing will start. In a carefully controlled test, it is expected that the input data stream will not be started until the processor has given a prompt indicating that it is ready.

2.2.2.2 Restart. This command is the same as the “reboot” command except that the setup constants will be left alone and the selftest will not be run.

2.2.2.3 Init. Inputs the following two words of setup information, in the order given below:

1. An integer between 0 and 15 which determines which polarizations are processed. For the purpose of controlling polarizations this integer should be thought of as 4 binary bits (although it is input as a decimal integer). Each bit enables the processing of a polarization as follows: VV, VH, HV and HH respectively where the MSB enables the processing of VV and the LSB enables the processing of HH. For example the number 5 would enable the processing of VH and HH.
2. The number of taps in the FIR filter used to convert input video to baseband I/Q. This will be an integer between 8 and 48.

If no “Init” command is given the default shall be: (1) to process only the HH polarization, corresponding to a value of 1 for the first word; (2) an FIR filter length of 8.

2.2.2.4 Run. This command enables the processing of data. If a data stream is already coming in, the SAR processor will start processing it with the start of the next PRI—defined by the next Barker sequence associated with data for the HH polarization. If there is no data stream coming in the processor will start processing whenever a data stream starts coming in (once the processor is ready). The processor indicates that it is ready to process any incoming data by giving a prompt in response to this command. If subsequent run commands are given, when the processor is already enabled for processing, the processor will respond with a prompt, leaving processing enabled.

2.2.2.5 Stop. This command disables processing. If it is given while data is coming in, processing will continue until the end of the current frame. It is OK if this is the next frame instead of the current one.

2.2.2.6 Status. In response to the status command the processor will respond with status information. The format of this information will be left up to the Developers—each Developers may implement the status checks and messages differently. The status information will not exceed 20 lines of 80 characters each. At a minimum the status information shall include the following:

1. Which polarizations are being processed.
2. How many taps are being used in the video to baseband FIR filter.
3. Whether the processor is enabled for processing.
4. How many frames have been processed since the last run command.

2.2.2.7 Step. Run for a single frame and then stop.

2.2.2.8 StepN. This command takes an integer between 1 and 32767 as the value of N. It runs for the number of frames specified by N.

2.2.2.9 Debugger. This command will cause control of the RS-232 line to be transferred to a debugger. It is left to the discretion of the Developers as to the commands for the debugger. It is expected that the capability will exist for examining and writing to memory locations within the SAR processor. When the debugger is entered, a message should be output on the RS-232 line saying that the debugger is being entered and what the command is to get back to normal command mode, where commands can once again be given to the signal processor.

2.2.2.10 Loadref. This command loads the reference kernels. The reference kernels are range-dependent matched filters which are convolved with the radar pulse returns to yield the desired images. The kernels are precomputed for a given range swath in the host and downloaded to the SAR image processor prior to operation. Detailed descriptions of the kernel and weighting functions are included in Appendices A and B.

The 31 kernels are loaded in sequence from near range (*kernel0*) to far range (*kernel30*), where each kernel is a vector of complex numbers. The elements of each kernel are loaded in order from index 0 to index 1023 (see Figure 9), and each element is composed of a real component followed by an imaginary component.

2.2.2.11 Loadequal. This command loads the equalization weights._Equalization weights change slowly and can be assumed constant over a data collection period. Therefore the equalization weights are precomputed and downloaded to the SAR processor prior to operation.

Equalization weights for the HH, HV, VH, and VV are loaded in order where the weights for each polarization are an array of complex numbers. The elements of each array are loaded in order from index 0 to index 2047, where the indices correspond to the ordered output samples of the I/Q filters (see Section 2.1.3). Each array element is composed of a real component followed by an imaginary component.

2.2.2.12 Loadiq. *Loadiqeven* and *Loadiqodd* commands load the I/Q filter weights. These weights are precomputed and downloaded prior to operation. Presently, 8-tap FIR filters are used but FIR filters with as many as 48 taps shall be accommodated. The number of real weights input by this command is always equal to 48, but only the first N weights are used where N is the number of taps specified by the last "Init" command. The even and odd filter weights are loaded in order from index 0 to index 47. The indices of the filter weights correspond to the FIR processing described in Section 2.1.3.

2.2.2.13 Loadrcs. This command loads the RCS weights._Weighting is applied to compensate the amplitude variations caused by beam-shape modulation in elevation and R^4 losses. These weights are computed in the host and downloaded prior to operation. The real-valued RCS weights are loaded in order from index 0 (i.e., near range) to index 2047 (i.e., far range)

2.2.2.14 Dump Commands. The commands "dumpinit", "dumpref", "dumpequal", "dumpiqeven", "dumpiqodd", and "dumprcs" will dump the parameters specified earlier by the corresponding init/load commands. The order of the numbers output will be the same as it is for input.

2.2.2.15 Selftest. Run one pass of the self test diagnostics. This is intended to run the same diagnostics that are run at system boot time. If this is not practical, a reasonable set of diagnostics will be run instead. If the diagnostics pass, they will output the line "Self test passed". If there are any errors, an appropriate error message will be given. The first character in a line corresponding to an error message will be a "?".

2.2.2.16 SelftestN. Run N passes of the self test diagnostics, where N is specified as an input parameter. N is an integer between 1 and 32767.

2.2.2.17 Linktest. Run one pass of a test of the fiber-optic interfaces. This test assumes that the fiber-optic interface is looped by locally, without actually going through a fiber. This test will transmit a pattern and then verify that it is received on the HotRod receiver. The pattern(s) used are left to the discretion of the Developers. This test can be run without the need to move any cables. "Linktestf" described below will require the disconnection of the normal fiber-optic cables and a connection of a loopback cable.

2.2.2.18 LinktestN. Run N passes of the "linktest", where N is specified as an input parameter. N is an integer between 1 and 32767.

2.2.2.19 Linktestf. Run one pass of a test of the fiber-optic interfaces. This test is the same as "linktest", described above, except that it requires a fiber-optic loopback cable—a cable which connects the SAR processor's fiber-optic transmitter to the fiber-optic receiver.

2.2.2.20 LinktestfN. Run N passes of "linktestf", where N is specified as an input parameter. N is an integer between 1 and 32767.

2.2.3 Power Reset

In addition to the software reboot command implemented through the control and diagnostic interface, a hardware reset capability shall be included. Operation of the hardware reset button will cause the SAR processor to execute the normal power-on start-up sequence. This start-up sequence will be functionally the same as the software reboot command.

2.3 Form Factor Constraints

2.3.1 Size

The maximum allowable dimensions for the processor are 10.5" in height, 20.5" in length, and 17.5" in width. These dimensions encompass the chassis, cooling fans, power supply and cable headers and are consistent with the use of the processor on board a small unmanned air vehicle (UAV) such as the Leading Systems *Amber* UAV pictured in Appendix D.

2.3.2 Occupancy and Scalability

The functional requirements included in this statement of work only encompass the processing through the step of image formation. In the future, additional functionality may be required, such as automatic target recognition and image compression. The processor architecture should be scalable to support at least twice the processing and twice the aggregate communication bandwidth as that implemented in the initial configuration.

In addition to expansion space for twice the computational throughput and communication bandwidth, space must be available in the processor box for the addition of a 4-slot 6U VME chassis. The provision for a VME chassis, or availability of four contiguous slots in an existing internal chassis, is to enable the use of commercial VME boards for display or subsequent processing of the images.

2.3.3 Weight

The weight for a fully-loaded chassis, including the 4 slot VME chassis, shall be less than 60 pounds.

2.4 Environmental

Air cooling in an non-condensing environment is assumed. The temperature range of the ambient air will be 0^{o} C to 40^{o} C.

Due to cost considerations, shock and vibration testing will not be performed. However, the processor should be designed so that it can be operated on-board the ADTS aircraft in a user-supplied shock-mounted tray. Measured vibration and estimated crash loads for shock-mounted equipment on the Gulfstream aircraft are given in Table 8.

Table 8. Vibration and shock loads.

Parameter	Value
Flight Load - Worst Case	
Vertical	8.8 G
Longitudinal	6.0 G
Transversal	2.0 G
Random Vibration	0.8 G RMS
Crash Safety	15 G, 11 ms

2.5 Power Supply

The power supply will operate with an input voltage of 24 to 32 volts DC. The average[2] input power shall not exceed 500 watts in the baseline system.

Provision to handle a fully populated chassis should be made with the input power not to exceed 750 watts. The power supply need not be sized to handle the fully populated chassis (750 watts input) provided there are provisions to incrementally add modules to the initial supply. The power supply should conform

2. The average as computed over any .5 second interval.

to the requirements of MIL-STD-704D for input transients and MIL-STD-461C for conducted and radiated emission susceptibility.

2.6 Fault Detection, Isolation and Testing

2.6.1 Fault Coverage and Isolation

The fault coverage and isolation achievable in the processor design will be influenced by the level of integration of the COTS hardware employed. For example, a system assembled from COTS chips typically affords more opportunities for introducing built-in test capability than a system assembled from COTS board products. Given the limited time duration and cost constraints of Benchmark-1 it is difficult to establish, apriori, levels of fault coverage and isolation that will be achievable through available COTS hardware.

However, it is important to demonstrate that the RASSP design methodology addresses reliability issues. Therefore, a goal of 90 percent fault coverage is established for the processor. As discussed in Section 2.6.3, the appropriate degree of fault isolation is dependent on the maintenance concept chosen and on the cost to repair versus replace a given isolation group. The goal for isolation is to isolate faults to a level at which replacement of the faulty group is commensurate in cost with replacement.

These fault coverage and isolation goals are included to demonstrate a capability to deal with design for test and built-in test in the RASSP process. The goals are not intended to become the major cost drivers of the design.

2.6.2 Data Stimulus

In addition to whatever test vectors and test patterns the Developer may provide in connection with the requirements of Section 2.6.1, the Benchmarker will supply a stimulus dataset consisting of unprocessed ADTS data, along with a set of reference images formed from the data using the algorithms described in Section 2.1. This data will be provided on an 8mm Exabyte 8200 or Exabyte 8500 tape cartridge in uncompressed tar format.

2.6.3 Maintenance and Testability

Testability requirements are a function of the maintenance concept chosen to support the deliverables of a particular benchmark. The choice of maintenance concepts is limited for a COTS implementation, whereas many maintenance concepts are applicable for custom implementations. In either case, the maintenance concept chosen must provide adequate support for the defined benchmark. Although no hardware deliverables are required for Benchmark-1, the maintenance concept and corresponding software simulation of hardware must be suitable for a military UAV application. Diagnostics must be run and errors must be reported via the control and diagnostic interface.

Testability of a COTS implementation is limited by the design of the COTS elements as well as any COTS building blocks used to integrate the elements. At a minimum, comprehensive built-in diagnostic capabilities (e.g., power-up memory tests) provided by the COTS product manufacturer must be fully utilized, and any errors must be reported by the system through the diagnostic and control interface. Test capabilities built into the COTS elements must be fully accessible from the control and diagnostic interface. If COTS elements with suitable capabilities are not available, then semi-custom diagnostics must be provided by the COTS manufacturer at the expense of the developer (e.g., a custom PROM in the case of hardware computer boards), and/or the developer must provide semi-custom integration elements which provide the necessary test capabilities (e.g., a COTS interface board with custom software). At a minimum, fault isolation must be to the COTS element level, and preferably beyond if such fault isolation capability is provided by the manufacturer of the COTS element. For example, if COTS boards are used (or simulated in software) then fault isolation must be at least to the board level, and may be to the chip level if such diagnostic capabilities are available for the COTS boards.

Many maintenance concepts are applicable for custom designs, and the testability requirements are chosen to be commensurate with the maintenance concept. An applicable maintenance/sparing concept which minimizes the estimated life cycle cost may be chosen by the developer. Parametric life cycle cost estimation programs ([9], [10]) for both hardware and software can be used to guide the choice. For example, the PRICE HL program estimates life cycle costs for 28 different standard maintenance concepts (e.g., discard line replacement unit at failure, replace module at organization and scrap bad module, replace module at equipment and repair module at contractor). Any of the maintenance concepts suitable for a military UAV are acceptable.

2.6.4 Acceptance Testing

Acceptance testing will consist of demonstrating reliable operation of the virtual prototype. The data will be supplied to, and collected from, the virtual prototype using the test bench supplied by the Benchmarker. Successful execution of all of the control and diagnostic modes indicated in Table 7 will be demonstrated as part of the acceptance testing.

2.7 Design Trades

The Developer must evaluate at least two architectures in terms of the following criteria:

1. Adherence to the requirements provided in Section 2.1 through Section 2.6, including the fault coverage and isolation goals of Section 2.6.1
2. Cost to produce in prototype quantities; essentially non-recurring engineering cost
3. Life cycle cost assuming a production of 500 units
4. Size, weight and power, with the emphasis first on low power and second on low weight

The basis for estimating the life cycle cost should be clearly described. The Developer may select the preferred architecture to develop as a virtual prototype either on the basis of low cost or on the basis of low power and weight. The intent is to allow the Developer to select the architecture which affords the best opportunity to demonstrate aspects of the virtual prototyping methodology, and will lead to a successful hardware fabrication in Benchmark-2.

2.8 Documentation

2.8.1 Virtual Prototype

A complete set of drawings shall be provided with each virtual prototype of the SAR image processor. For parts of the prototype processors that are COTS (commercial off the shelf), some of these requirements may be waived. At a minimum, the drawings must include both simplified and detailed block diagrams. Depending on the detail of the design prototype achieved during the benchmark cycle, additional drawings shall include, but not be limited to, the following:

- Individual mechanical drawings of chassis, boards, backplanes and connectors.
- Detailed schematics and/or source files for all non-COTS printed circuit boards, MCMs, ASICS, PALs, FPGAs and PLDs.
- All source files and/or schematics for any programmable devices incorporated in the signal processor, including PALs, FPGAs, and complex PLDs. This requirement is for the lowest level description that was used in the course of designing the device.
- Parts list.
- Net list of all non-COTS printed circuit boards.
- Full Specifications for any non-standard or proprietary components.

The theory of operation shall be documented including,

- Modes of operation supported and the protocols for the test and diagnostic defined in Table 7.
- All critical timing information.
- All non-standard interfaces.

2.8.2 Software

All non-COTS application software (i.e software developed specifically for the Benchmark by the Developer) shall be provided in Ada. Wherever available, COTS application source code shall be provided in Ada. Hard copy of all application source code shall also be provided. The intent here is that Ada should be used except where significant reductions in performance or increases in cost would result. An example would be the case where an Ada development environment does not exist for the target processor.

Software documentation shall conform to best commercial standards and practices.

2.9 Reporting

Progress reports shall be provided with each milestone as discussed in Section 5.

3. EXECUTABLE REQUIREMENTS

MIT Lincoln Laboratory, at the direction of the Government, will deliver to the Developers an executable requirement consisting of a VHDL behavioral model of the SAR processor as specified in Section 2, a VHDL test bench, and input stimulus and output comparison data files. The delivery will include a User's Manual.

3.1 Overview

The SAR Processor and Test Bench models shown in Figure 12 are implemented in IEEE 1076-1987 compliant VHDL in a manner which should make them executable with any compliant simulator. The VHDL executable requirement was originally developed on a Vantage simulator and ported to Mentor QuickVHDL. If further changes are required to run on the Developer's simulator, Lincoln Laboratory and its subcontractor(s) shall be given access to the simulator for the purpose of making any required changes.

In the VHDL executable requirement, the interfaces between the test bench and behavior model of the processor, for both data input and data output, are implemented at the processor parallel interface of the TriQuint HRC-500FS module described in Appendix C. For reasons of efficiency, the Control and Diagnostic Port is implemented as a 32-bit interface between the processor and test bench rather than the RS232 interface required in the actual processor.

3.2 Processor

The VHDL model of the SAR processor models timing behavior only for data streams at the Input Port and Output Port. Processing is done in zero simulation time. The model is built with sufficient internal buffering so that latency from the beginning of the 512th PRI input data set of a range-azimuth frame and the first output data set for that frame can be set from 0.1 seconds to 3 seconds. The model implements the following commands from Table 7: Reboot, Restart, Init, Step, StepN, Loadref, Loadequal, Loadeqeven, Loadiqodd and Loadrcs.

3.3 Test Bench

The VHDL Test Bench reads control, data, setup and comparison files from disk and writes output and log files to disk. The test bench is controlled from a text script with the Control and Diagnostic Port commands listed above. The test bench compares the processor output data against supplied comparison files using the criteria of Section 2.1.1. The output of the comparison is the number of pixels with error exceeding the specification, which is set with a VHDL generic, and the index and error value of the pixel with largest error. Latency is measured and compared with a limit which is set with a VHDL generic. Comparison results are written to the log file. The test bench may be modified to accommodate a Control and Diagnostic Port of the Developer's design. It may also be modified to implement the Dump and Status commands of Table 7 when it is appropriate to do so. Modifications will be performed as a coordinated effort between Lincoln Laboratory and the Developers.

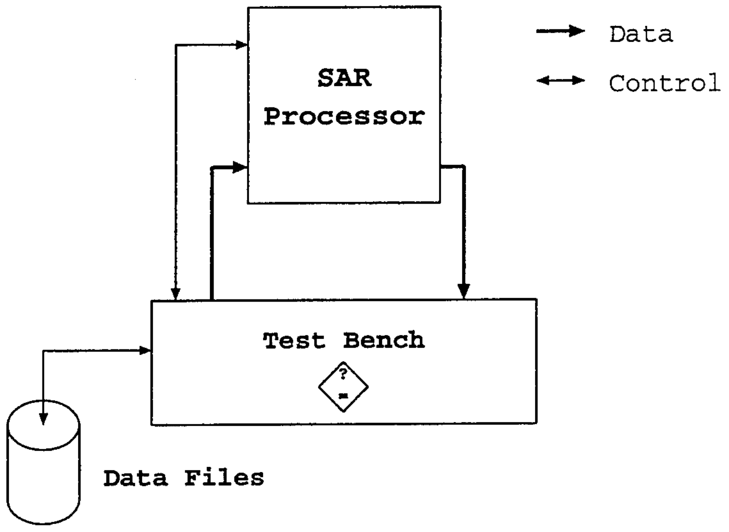


Figure 12. Processor Model.

3.4 Data Files

All data files are supplied in a format readable by the Test Bench. Input data files from the ADTS will be supplied for four frames of data. A synthetic data set will also be supplied and it may be possible for MIT Lincoln Laboratory to create additional synthetic data sets at the Developer's request. Setup files will be supplied. Reference image data sets which have been generated by a C-language program from the supplied input data and setup data will be supplied. The entire set of data may occupy more than 1 GByte of disk space.

3.5 Auxiliary Software

MIT Lincoln Laboratory will also provide programs which convert the output files to an intermediate form and display images. The programs are written in C and have been compiled and executed in SunOS 4.1. The display program uses the XllR5 interface. Programs to perform comparisons between different output data sets in this intermediate format will also be provided.

4. METRICS

4.1 Introduction

All metrics associated with Benchmark-1 are described in this section. However, not every metric identified in this section will necessarily be used in the Benchmark-1 Evaluation Report. Additional metrics may also be devised as Benchmark-1 progresses. The metrics which are currently believed to be essential for developing a comprehensive evaluation report are identified in Section 5 as deliverables which the Developer must collect and supply to the Benchmarker during the course of Benchmark-1 execution. In some cases, only estimates of the required metrics or parameters will be available. In such cases, the Developer will supply a best estimate with a rationale (basis) for the estimate.

Two approaches will be applied to evaluate the RASSP process and products. In the first approach, commercially available parametric cost estimation packages will be utilized, primarily to obtain estimates of cost and schedule to serve as the current practice reference. The most comprehensive packages are the Parametric Review of Information for Costing and Evaluation (PRICE) and System Evaluation and Estimation of Resources (SEER). These packages are discussed in Section 4.2. In a second approach, metrics derived from basic principles will be collected and utilized as a basis for evaluating specific areas of RASSP product and process development. Such development areas include productivity measures of the RASSP process such as lines of code per day produced, ease of use of the design environment, performance and complexity of the product, quality of the product, cost of the process and the product. The metrics formulated for these and other areas are discussed in Section 4.3.

As the RASSP program develops, redundancies between the metrics derived from the parametric cost estimators (Section 4.2) and the process and product metrics (Section 4.3) will be identified and eliminated. Metrics that do not correlate with the observed performance of the RASSP process and products will be modified or replaced. In this way, the set of metrics will be continually refined over the duration of the RASSP program.

Since a primary goal of the benchmark activity is quantitative measurement of RASSP related improvements to design, it is anticipated that the collection and analysis of metrics for this purpose will require a non-trivial effort on the part of the Developers and the Benchmarker. In formulating the Benchmark Execution Check List, the Developer should indicate the estimated cost to collect the metrics identified in Section 5.2. according to the breakdown of deliverables provided in Table 18 through Table 30.

4.2 Parametric Cost Estimators

Progress is measured in terms of costs, which are defined in the general sense where expense, development time, and manpower requirements are considered to be "costs." There are many software tools for cost tracking and estimation, and the benchmark evaluation approach involves several such tools.

A technique called "detailed bottom-up estimating" [6] is used for cost tracking purposes. This method is analogous to a bill of materials and labor required to produce each subassembly in a system, and the cost of integrating the subassemblies. Actual costs collected from the vendors are entered into a computer system using a Microsoft Excel spreadsheet program, which has a data format compatible with all of the parametric cost estimation programs discussed later. This approach allows data to be conveniently entered, organized, analyzed and updated. Data accumulated from the various Developers and their subcontractors at different phases of the benchmarks are archived in a database for future reference. Effects such as the normal progress of technology in the absence of RASSP can be factored out of the database at a later time, if desired. The cost tracking feature of the Microsoft Project planning program is used in conjunction with the database to determine the incremental and overall rate of progress in all areas of interest.

In a cost estimation technique known as "parametric estimating," a cost estimating relationship (equation, table or graph) is used to predict cost as a function of design size, performance variables, applicable technology and other parameters. The Air Force provides a free program called REVIC which performs software cost estimates based on the Constructive Cost Model (COCOMO) [7]. In addition, there are at least 18 commercial companies which provide parametric cost estimation products for software [8]. Two product lines (PRICE from Martin Marietta PRICE Systems and SEER from Galorath Associates Inc.) are of particular interest as they also provide hardware cost estimation capabilities. These programs require a variety of inputs to perform their cost estimation function. The inputs to these various cost estimation programs form a basic set of metrics which can be used to track the progress of RASSP, and other metrics can be added as necessary. Note that actual benchmark measurements, not the predictive cost estimates produced by the programs, will ultimately measure the progress. The cost estimates produced by the programs can, however, be used to compare the complexity of one benchmark task relative to another benchmark task. In addition, the cost estimates can be used to identify areas in which progress is being made (e.g., a measured cost which is less than the current practice-based predictive cost estimates by a factor of 4 indicates potential achievement of a RASSP goal in a particular area).

4.2.1 COCOMO -- Constructive Cost Model

One set of metrics for tracking the progress of RASSP software is provided by the U.S. Air Force's COCOMO-based program, REVIC. Because the inputs required by REVIC are a subset of those required by SEER (see Section 4.2.6 and Section 4.2.7), the REVIC inputs are not separately required deliverables. REVIC is discussed here for tutorial purposes, since it may be used by the Benchmarker in the evaluation process. This section contains an abbreviated description of the 17 software metrics, or cost drivers, used by COCOMO [7]. Note that these metrics must be applied in a framework which considers the development mode (ranging from a small straight-forward project to a large project requiring much innovation) and phase of the project (requirements, product design, detailed design, code and unit test, integrate and test, and maintenance). COCOMO makes estimates in terms of required effort (measured in staff months) and project duration (measured in months), while using different effort multipliers to modify the cost drivers according to project development mode and phase of development.

Incremental Development COCOMO is a modern spiral-model alternative to the traditional waterfall model used in the standard software development process modeled by COCOMO. Instead of modeling software development as if it were a single effort devoted to inventing a single product, Incremental Development COCOMO models development as a series of concurrent software projects, each yielding an intermediate product. This strategy reduces risk, and permits early delivery of an initial product. Although this feature does not affect the metrics, it is an important point to consider when selecting a COCOMO-based RASSP tool.

4.2.1.1 Definitions.

4.2.1.1.1 Virtual machine. For a given software product, the underlying virtual machine is the complex of hardware and software (operating system, database management system, etc.) it calls upon to accomplish its tasks.

4.2.1.1.2 Delivered source instructions. Delivered source instructions (DSI) are lines of source code delivered as part of the product. Test drivers and other support software are excluded. Source lines are created by the project staff, and code created by applications generators is excluded. One instruction is one line of code. Declarations are counted as instructions, while comments are not. The unit of measure is thousands of delivered source instructions (KDSI).

4.2.1.2 Product attribute software metrics.

4.2.1.2.1 RELY: required software reliability. A software product possesses reliability to the extent that it can be expected to perform its intended functions satisfactorily. Quantitatively, reliability is the probability that the software performs its intended functions satisfactorily over its next run or its next quantum of execution time. Reliability can be calculated if the software's operational profile (the probability distribution over the space of possible inputs or input sequences to the software, representing the probability that each input or input sequence will be selected for the next run or quantum of execution time) is known, and if a precise definition exists for what it means when the software "performs its intended functions satisfactorily." Given this information, a minimum variance unbiased estimator of reliability can be calculated by first choosing N inputs or input sequences selected at random from the operational profile distribution, using the inputs to exercise the software for N runs or execution time quanta, use the success criterion to determine how many runs or quanta resulted in M satisfactory outcomes, and computing M/N.

While this definition formalizes the concept of software reliability, it is of limited value in practice due to the difficulty in determining the probability distributions over the space of possible inputs. In practice, reliability is often characterized through a defect rate per line of code.

4.2.1.2.2 DATA: database size. The total amount of data to be assembled for the database (measured in bytes or characters) divided by the program size (measured by the number of delivered source instructions in the software product, KDSI) is used to incorporate database considerations into COCOMO.

4.2.1.2.3 CPLX: product complexity. The complexity is judged to be in one of the following six categories, and a multiplier is chosen according to the project phase.

Very low	straight-line code, evaluation of simple expressions, I/O statements with simple formats, simple arrays in main memory.
Low	straight-forward nesting of operators, evaluation of moderate-level expressions, no knowledge of I/O device characteristics, single-file subsets with no data structure changes.
Nominal	Simple nesting with some intermodule control and decision tables, use of standard math with matrix and vector operations, I/O device selection with status checking and error recovery, multifile input and single file output.
High	highly nested operators with queue and stack control, considerable intermodule control, numerical analysis including multivariate interpolation, differential equations, roundoff error concerns, operations at physical I/O level (including physical storage address translations and seeks), optimized I/O overlap, special-purpose subroutines activated by data stream contents, complex data restructuring at a record level.
Very high	reentrant and recursive coding, fixed-priority interrupt handling, difficult but structured numerical analysis, near-singular matrix equations, partial differential equations, routines for interrupt servicing and masking, communication line handling, a generalized parameter-driven file structuring routine, file building, command processing, search optimization.
Extra high	multiple resource scheduling with dynamically changing priorities, microcode-level control, difficult and unstructured numerical analysis, highly accurate analysis of stochastic data, device timing-dependent coding, microprogrammed operations, highly coupled dynamic relational structures, natural language data management.

4.2.1.2.4 REUSE: required reusability. The reusability is judged to be in one of the following four phases, and a multiplier is chosen according to the project phase.

Nominal	Not for reuse elsewhere.
High	Reuse within single-mission products.
Very High	Reuse across a single product line.
Extra High	Reuse in any application.

4.2.1.3 Computer attribute software metrics.

4.2.1.3.1 TIME: execution time constraint. One of four categories is chosen depending on the percentage of available execution time to be used by the subsystem and any other subsystems consuming the execution time resource, and a multiplier is chosen according to the project phase.

Nominal	up to 50%.
High	51-70%.
Very high	71-85%.
Extra high	above 85%.

4.2.1.3.2 STOR: main storage constraint. One of four categories is chosen depending on the percentage of main storage to be used by the subsystem and any other subsystems consuming the main storage resource, and a multiplier is chosen according to the project phase. Main storage refers to high-speed memory from which programs are executed, not to disk capacity.

Nominal	up to 50%.
High	51-70%.
Very high	71-85%.
Extra high	above 85%.

4.2.1.3.3 VIRT: virtual machine volatility. This cost driver expresses the effects of changes in the underlying virtual machine for which the software is being developed. A major (minor) change is one which significantly effects roughly 10% (1%) of routines under development. The VIRT value is low for a major (minor) change frequency of 12 months (1 month), nominal for 6 months (2 weeks), high for 2 months (1 week) and very high for 2 weeks (2 days).

4.2.1.3.4 TURN: computer turnaround time. The values for this variable are determined by the average time from data processing job submission until results are returned. The TURN value is low for interactive systems, nominal for systems requiring up to 4 hours, high for systems requiring 4 to 12 hours and very high for systems requiring over 12 hours.

4.2.1.4 Personnel attribute software metrics.

4.2.1.4.1 ACAP: analyst capability. This variable expresses the rating for an analyst team with regard to analysis ability, efficiency, thoroughness, and ability to communicate and cooperate. Experience is not a factor (see AEXP). Ratings are expressed as a percentile in comparison to all other analyst teams.

The specific ratings are:

Very low	15%
Low	35%
Nominal	55%
High	75%
Very high	90%).

4.2.1.4.2 AEXP: applications experience. This variable expresses the level of applications experience of the project team developing the software subsystem. The ratings are defined in terms of the team's level of experience with this type of application:

Very low	<4 months average experience
Low	1 yr.
Nominal	3 yrs.
High	6 yrs.
Very high	>12 yrs., or reimplementation of subsystem

4.2.1.4.3 PCAP: programmer capability. This variable expresses the rating for a programming team with regard to programming ability, efficiency, thoroughness and ability to communicate and cooperate. Experience is not a factor (see VEXP). Ratings are expressed as a percentile in comparison to all other programming teams. The specific ratings are:

Very low	15%
Low	35%
Nominal	55%
High	75%
Very high	90%

4.2.1.4.4 VEXP: virtual machine experience. This variable expresses a rating for the level of virtual machine experience of the project team, excluding programming language (see LEXP). Ratings are defined by the project team's average experience with the virtual machine to be used:

Very low	<1 month
Low	4 months
Nominal	1 yr.
High	>3 yrs

4.2.1.4.5 LEXP: language experience. This variable expresses the project team's average duration of experience with the programming language to be used:

Very low	<1 month
Low	4 months
Nominal	1 yr.
High	>3 yrs.

4.2.1.5 Project attribute software metrics.

4.2.1.5.1 MODP: modern programming practices. Modern programming practices such as top-down requirements analysis and design, top-down incremental development, structured design notation and code, design inspections (code walkthroughs) and assignment of a program librarian (configuration control) play an important role in productivity. This variable expresses use of modern programming practices with the following ratings:

Very low	no use
Low	beginning or experimental use
Nominal	experienced in use of some
High	experienced in use of most
Very high	routine use of all modern programming practices

4.2.1.5.2 TOOL: use of software tools. Software tool type, quality and degree of integration play a major role in productivity. This variable expresses the availability of software tools rated as follows:

Very low	basic microprocessor tools
Low	basic minicomputer tools
Nominal	strong minicomputer or basic maxicomputer tools
High	strong maxicomputer tools
Very high	advanced maxicomputer tool

4.2.1.5.3 SCED: required development schedule. The software development effort is a function of schedule constraints imposed on the project team. Ratings for this variable are defined in terms of the percentage of schedule stretch-out or acceleration with respect to a nominal schedule (which is in turn a function of the project development mode and phase of development). Only a limited range of percentages are considered:

Very low	75%, severe acceleration
Low	85%, moderate acceleration
Nominal	100%
High	130%, moderate stretch-out
Very high	>160%, severe stretch-out

4.2.1.5.4 SECU: classified security application. The level of classification is judged to be either of the following two phases, and a multiplier is chosen according to the project phase.

Nominal	Unclassified.
High	Classified (Secret, Top Secret)

4.2.1.6 Factors not included in standard COCOMO. Factors such as type of application (control vs. algorithm), language level (delivered source instructions vs. deliverable executable machine instructions), other size measures (complexity, program entities such as routines or files, and number of paragraphs in the software requirements specification), requirements volatility (amount of change in software requirements between beginning and end of a project), personnel continuity (turnover), management quality (failure to prepare resources), customer interface quality (poor communications), amount of documentation (large amounts of poor quality documentation vs. smaller amounts of good), hardware configuration (effects of poor support and reliability) and security (privacy) restrictions are not specifically included in the standard COCOMO model. Many of the effects of these factors, however, are already covered by other factors in COCOMO.

4.2.2 PRICE S software

The PRICE Software Model applies parametric modeling methods to estimate the acquisition cost, software sizing cost, and operating and support costs for computer software. The acquisition cost estimates the software development acquisition process in each of the following phases:

1. System concept
2. System software requirements
3. Software requirements analysis
4. Preliminary design
5. Detailed design
6. Computer software configuration item (CSCI) test
7. System test
8. Operational test and evaluation (OTE)
9. System integrate and test.

The software sizing cost estimates the number of instructions in terms of source lines of code for both commercial and military applications. The operating and support costs estimate the life cycle costs for the maintenance phase, including software maintenance, enhancement, growth, and modification.

4.2.2.1 Acquisition Mode. For the acquisition mode, cost estimates are made using an EBS (Estimating Breakdown Structure) which is a sideways tree structure that provides a graphical, hierarchal representation of the system to be estimated. Associated with the element at the system level are the output, global (includes schedule multipliers, cost element multipliers, sensitivity step variables, person-hours per month, person-month decimals), financial factors (includes element labor rates, overhead, cost of money rates, overtime percentage, general and administrative rates, profit percentage, economic base year, escalation on/off), escalation (includes inflation rates from 1946-2025), and deployment data types which allow for further customization of the cost estimate. Each subordinate element in the tree has an associated data type of one of eight categories--Development CSCI, Purchased CSCI, Furnished CSCI, Calibration CSCI, Development CSC (computer software component), Purchased CSC, Furnished CSC, and Language--which each has its own specific variable inputs. Some of these inputs appear in more than one category.

4.2.2.1.1 Development CSCI.

PLTFM	platform; the customer's requirements stemming from the planned operating environment; measures acceptability of portability, reliability, structuring, testing and documentation.
CPLXM	management complexity; effect of complicating factors (e.g. development on a multinational level or at more than one location)
INTEGI	internal integration; level of internal integration of lower level work packages of the CSCI.
INTEGE	external integration; level of integrating CSCIs into the next higher level system.
UTIL	utilization; the fraction of available hardware cycle time or total memory capacity used.
SCON	the date the system concept effort starts
SDR	the date the system design review is complete or,
SSR	the date the software specification review is complete.
SRR	date the system requirements analysis review is complete
PDR	date the preliminary design review is complete
CDR	date the critical design review is complete
TRR	date the test readiness review is complete
FCA	date the functional configuration audit is complete
PCA	date the physical configuration audit is complete
FQR	date the formal qualification review is complete
OTE	date the operational test and evaluation is complete

4.2.2.1.2 Purchased CSCI.

LANG	source langiage; source language of purchase s/w equipment.
SLOC	source lines of code; total number of SLOC to be purchased
FRAC	fraction of non-executable code; fraction of SLOCs describing the type declarations and data statements.

APPL	application; expression of the application mix of instructions--low values correspond to math and string- manipulation; high values emphasize Real-Time Command and Control and interactive applications.
INTEGE	see Section 4.2.2.1.1
PCOST	cost of purchased component; cost of purchased software
UNITS	cost units; provides the unit of measurement (hours, months, currency) for the PCOST input
RATE	cost of labor for the development of the purchased s/w.
RATE TIME UNIT	time per hour or per month used for the RATE input.
PLTFM	same as in Development CSCI.

4.2.2.1.3 Furnished CSCI.

LANG	see Section 4.2.2.1.2
SLOC	see Section 4.2.2.1.2
COST	the cost of the software to be calibrated
FRAC	see Section 4.2.2.1.2
APPL	see Section 4.2.2.1.1
INTEGE	see Section 4.2.2.1.1
PLTFM	see Section 4.2.2.1.1

4.2.2.1.4 Calibration CSCI. Runs price backwards using performance characteristic of recent previous projects to calibrate the current cost estimation.

PLTFM	see Section 4.2.2.1.1
CPLXM	see Section 4.2.2.1.1
INTEGI	see Section 4.2.2.1.1
UTIL	see Section 4.2.2.1.1
COST	see Section 4.2.2.1.1

SDR or SSR	see Section 4.2.2.1.1
SCON	see Section 4.2.2.1.1
SRR	see Section 4.2.2.1.1
PDR	see Section 4.2.2.1.1
CDR	see Section 4.2.2.1.1
TRR	see Section 4.2.2.1.1
FCA	see Section 4.2.2.1.1
FQR	see Section 4.2.2.1.1
OTE	see Section 4.2.2.1.1

4.2.2.1.5 Development CSC

INTEGI	see Section 4.2.2.1.1
INTEGE	see Section 4.2.2.1.1
UTIL	see Section 4.2.2.1.1
SSR	see Section 4.2.2.1.1
PDR	see Section 4.2.2.1.1
CDR	see Section 4.2.2.1.1
TRR	see Section 4.2.2.1.1
FCA	see Section 4.2.2.1.1

4.2.2.1.6 Purchased CSC

LANG	see Section 4.2.2.1.2
SLOC	see Section 4.2.2.1.2
FRAC	see Section 4.2.2.1.2
APPL	see Section 4.2.2.1.2
INTEGE	see Section 4.2.2.1.1
PCOST	see Section 4.2.2.1.2

UNITS	see Section 4.2.2.1.2
RATE	see Section 4.2.2.1.2
RATE TIME UNIT	see Section 4.2.2.1.2

4.2.2.1.7 Furnished CSC

LANG	see Section 4.2.2.1.2
SLOC	see Section 4.2.2.1.2
FRAC	see Section 4.2.2.1.2
APPL	see Section 4.2.2.1.2
INTEGE	see Section 4.2.2.1.1

4.2.2.1.8 Language

LANG	see Section 4.2.2.1.2
SLOC	see Section 4.2.2.1.2
FRAC	see Section 4.2.2.1.2
CPLX1	complexity 1; a quantitative description of the relative effect of complicating factors such as product familiarity, personnel skills, software tools, etc. on the software development task.
CPLX2	hardware/software interactions complexity; a quantitative description of the relative effect of complicating factors such as new hardware development and hardware developed in parallel caused by hardware/software interactions.
PROFAC	productivity factor; an empirically derived parameter that includes items such as skill level, experience, productivity, and efficiency.
APPL	see Section 4.2.2.1.2
NEWD	new design; the percentage of new design effort.
NEWC	new code; the percentage of new coding effort.

4.2.2.2 Software Sizing Mode. This mode estimates the number of instructions in SLOC needed for commercial and military applications.

4.2.2.2.1 Commercial

INTEGRATION	will the software program be integrated with other software programs?
DESIGN REVIEW	is an in-house or customer design review required?
CODE WALK-THROUGH	will the programmer have to walk through the program with peers and offer a forum for its discussion?
TOP DOWN APPROACH	will the top-down approach be used?
MODULE TESTING	is modular testing (build a little, test a little) required?
OUTP	output pages; the number of unique output pages directed to a line printer (one output page equals 66 lines)
OUTS	output screens; the unique number of format output reports or data format that will be output to a CRT display @ 24 lines/screen.
OUTD	output displays; the unique number of format graphic outputs that will be displayed on a CRT or plotting device.
INPF	input files; a quantitative description of the number of unique input streams to the software package.
OUTF	output files; a quantitative description of the number of unique output streams of the program
SCRF	scratch files; the number of temporary work or scratch files that will be used internally by the software program for temporary storage, calculations, etc.
COPT	control options; the number of control options or modes of operation of the software program.
INPFV	input variable/fields; a required input that describes to the model the number of different variable fields.
COMVA	computed or created variables; describes the number of created tables/ variables used by the software program for various calculations.
LANG	language; the source language to be used for the software development effort.
TARSIZ	target size; the number of SLOC from a completed software program for calibration purposes.

SICAL	the size calibration factor
REQG	requirements growth; anticipated growth from any uncertainty or room for revisions in the original system requirements stage.
FBULK	functional bulkiness; an efficiency rating of the software program that takes into account the programmer's skill and the effectiveness of available programming tools in minimizing the amount of instructions being written.

4.2.2.2.2 Military

MILITARY/ COMMERCIAL	accounts for the inherent complexity of the project by its specification level required with military projects generally being more specific and complex.
INTEGRATION	see Section 4.2.2.2.1
DESIGN REVIEW	see Section 4.2.2.2.1
CODE WALK-THROUGH	see Section 4.2.2.2.1
TOP DOWN APPROACH	see Section 4.2.2.2.1
MODULE TESTING	see Section 4.2.2.2.1
OUTP	see Section 4.2.2.2.1
ALPD	alphanumeric displays; a quantitative description of the number of unique alphanumeric display formats--similar to OUTP.
GRAFD	graphics displays; the unique number of operator graphic display formats for rasters or other types of graphic displays, X-Y plotting boards, and other real time command and control devices that employ designs using pixels, aspect ratios, etc.
INPST	input streams; software digital data signals received by a CSCI or CSC that contains unique address data that instructs the receiving module concerning the use of the message data contained on the stream.
OUTST	output streams; software digital data signals generated by a CSCI, CSC, or other piece of operating hardware such as servo mechanisms, printers, etc.

CSTATE	control states; major decision points in a software program that branch into two or more optional program routines.
INPMF	input message field; the portion of an INPST or OUTST that contains the intelligence being transmitted in the form of messages.
INPDK	operator actions; any operator activity that results in a digital signal being sent to a CSCI.
INPAN	input analogs; signals which are converted into digital signals prior to transmittal to the CSCI.
COMTA	computer or created tables; the number of data elements (digital words or acronyms) that are accumulated in tables (either in matrix form or in active storage.)
FBULK	see Section 4.2.2.2.1
REQG	see Section 4.2.2.2.1
SICAL	see Section 4.2.2.2.1
TARSIZ	see Section 4.2.2.2.1
LANG	see Section 4.2.2.2.1

4.2.2.3 Life Cycle (Operating and support) Mode. This mode estimates software maintenance, enhancement, growth, and modification costs.

PLTFM	see Section 4.2.2.1.1
UTIL	see Section 4.2.2.1.1
SSR	see Section 4.2.2.1.1
SCHFAC	schedule fraction; the amount of software development schedule acceleration or stretch out.
DEVCST	development cost; the total software development cost
DEVU	development units; the units (Hours, Months, Currency) entered for the DEVCST input.
RATE TIME UNIT	see Section 4.2.2.1.2
RATE	see Section 4.2.2.1.2

4.2.3 PRICE-M micro circuits and electronic assemblies.

PRICE-M consists of three modes: microcircuit, module, and database. The micro circuit mode emulates the procedures and processes involved in the design and fabrication of microcircuits. The module mode represents a computerized modeling technique designed to produce cost and schedule estimates associated with the design and production of modules, boards, or hybrids. The database mode allows the user to place frequently used components into files which are then specified as extra input files in the module mode. Indices are derived from the calibration process based on cost history. (*) indicates optional data.

4.2.3.1 Module mode inputs.

4.2.3.1.1 General A.

QTY	quantity of production modules
PROTOS	quantity of prototypes
LENGTH, WIDTH	length and width of module in inches
LAYERS	number of discrete layers
PLTFM	platform (commercial low, commercial high, military fixed, military mobile or high fixed or commercial aircraft, military aircraft, or manned space)
NAME	name of module

4.2.3.1.2 General B.

MBINDX	manufacturing index based on cost history
BTYPE	bard type (material and use -- e.g., standard, RF, microwave, or power)
BSIDE	board sides (component layers, not related to LAYERS above)
*BCOST	board cost
*PTYPE	package type (material and use, as BTYPE)
*PPINS	number of connections
*PCOST	package cost
*ATCOST	assembly and test cost

4.2.3.1.3 PRICE-H Interface.

QTYNHA	number of modules to be integrated and tested into the next higher level of integration
INTEGE	electronic integration
HSINT	hardware/software integration (based on amount of modifications, simplicity of interface, and importance of timing)
WEIGHT	weight (pounds)
VOLUME	volume (cubic feet)
BWT	board weight (pounds)
PWT	package weight (pounds)

4.2.3.1.4 Development.

ECMPLX	experience and qualifications of engineering team based on amount of modification, technology level, and personnel experience)
NEWDES	percent new design
*DESRPT	percent repetition in design effort

4.2.3.1.5 Development schedule.

DSTART	design start date
DFPRO	date of completion of first prototype
DLPRO	design end date or date of qualification of last prototype
DBINDX	development index based on cost history

4.2.3.1.6 Production schedule.

PSTART	production start date
PFAD	date of completion of first production unit
PEND	production end date
*MAUTO	level of automation of assembly and testing (based on automation process-none, semi, or full- and description of assembly, such as robot assembly, standard assembly, or all hand insertion)
MMAT	level of experience and maturity of manufacturing process (based on new, similar, or same process and description of difference in process)

4.2.3.1.7 Supplemental information.

YRECON	year of economics (refers to cost outputs)
YRBASE	base year of economics (refers to cost inputs)
*YRTECH	year of technology
AUCOST	average unit cost
ETCOST	engineering total cost
PRCOST	prototype cost

4.2.3.1.8 Component data.

CNUM	number of components
CNAME	component name
CELM	number of active components

CTYPE	type of component
CPKG	component packaging
CPINS	number of connecting pins and/or pads
CWT	component weight
CCOST	component cost

4.2.3.2 Microcircuit mode inputs.

4.2.3.2.1 General.

QTY	production quantity
PROTOS	prototype quantity
LENGTH, WIDTH	length and width of chip (mils)
PINS	number of pins
GATES	number of gates
XSTRS	number of transistors
CNAME	component name

4.2.3.2.2 Development.

DPLTFM	design platform (ground, mobile, airborne, or space)
SPLTFM	system platform
DINDEX	development index (based on speed in MHz and design technology)
CMPLX	engineering complexity (based on personnel experience and scope of design effort)
NEWCEL	percentage of library circuit cells or macros needed to be designed
DESRPT	percentage of design repetition
CADFAC	CAD factor (based on CAD features)
TERAT	number of design/prototype/test integrations

4.2.3.2.3 Production A.

PROFAC	production factor
MINDEX	manufacturing index
PKGFAC	packaging factor
SUBFAC	substrate factor
LOTQTY	total lot quantity
WSIZE	wafer diameter (mm)
FSIZE	feature size (microns)

4.2.3.2.4 Production B.

CPYLD	circuit probe yield
ASMYLD	assembly yield
OVLYLD	overall yield
MSKLVL	mask levels
DEFDEN	defect density
MAUTO	manufacturing automation
MMAT	manufacturing maturity

4.2.3.2.5 Development schedule.

DSTRT	development start date
PTSRT	prototype start date
PTEND	prototype end date
TSTEND	prototype test end date
DEND	development end date

4.2.3.2.6 Production schedule.

PSTRT	production start date

PPEND	pre-production end date
PEND	production end date

4.2.3.2.7 Supplemental information.

YRECON	year of economics
AUCOST	average unit cost

4.2.3.3 Database mode inputs.

PLTFM	platform
YRBASE	base year

Component data (see Section 4.2.3.1.8).

4.2.4 PRICE H hardware systems.

Cost estimates are made via an Estimating Breakdown Structure (EBS). The EBS is a sideways tree structure which provides a graphical depiction of the system to be estimated. Fourteen items called elements can be selected from the EBS for editing, copying, moving, deleting or processing. The 6 primary hardware operation elements are: system, assembly, electro/mechanical, structural/mechanical, modified and calibration. The 3 integrating operation elements are: design integration, hardware/software integration and hardware integration & test. The 5 specialized elements are: purchased, given cost, furnished, thru-put and multiple lot production. Four different types of data or operations may be associated with each element: input, output, global and escalation. Input variables, or metrics, may have a different definition and value for each element of the EBS. Input variables can be grouped into 11 categories as follows, but there is considerable overlap and interaction between the categories.

4.2.4.1 Project magnitude. The number of development and/or production units. Included in this category is the weight, volume and/or the electronic weight or packaging density of the assembly

QTY	number of production units
PROTOS	number of prototypes
PROSUP	prototype support
WT	total weight

WS	structure weight
WECF	weight of electronics per cubic foot
WSCF	weight of structure per cubic foot
VOL	total volume
USEVOL	fraction of total volume used by electronics

4.2.4.2 Customer specification and reliability requirements. The specification level, operating environment and reliability requirements associated with the end use of the product.

PLTFM	platform type (e.g. car vs. spacecraft)
MREL	mechanical reliability (estimated Mean Time Between Failures)
EREL	electronic reliability

4.2.4.3 Complexity of design. A measure of the effort's technology, producibility (material, machining and assembly tolerance difficulty, etc.) yield, platform and all labor required to produce the structural and/or electronic part of the assembly.

HYBRID	percentage for each type of electronics that consist of hybrids
IC	percentage for each type of electronics that consist of integrated circuits
LSI	percentage for each type of electronics that consist of large scale ICs (100-1K gates)
VLSI	percentage for each type of electronics that consist of very large scale ICs (1K-1M gates)
MCONST	a constant used to describe material and style
MEXP	raw material type code
MCPLXS	manufacturing complexity of the structure

MCPLXS is a function of precision of fabrication (PRECI), machinability of material (MI), difficulty of assembly (MATUR), number of parts (NP) and platform (PLTFM). Additional input parameters (e.g., HOGOUT if more than 10% of slug weight is machined away), including historical data, can also be applied.

MCPLXE	manufacturing complexity of the electronics

MCPLXE is a function of the type of electronics (analog, digital, display, etc.), electronic componentry (discrete devices, ICs, hybrids, etc.), specification (testing level varies with platform) and various adjustments (component quality, density and a calibration factor based on historical data).

Calibration procedures use actual cost data from completed projects to determine historical values for MCPLSX and MCPLXE. This operation is referred to as ECIRP, which is PRICE spelled backwards. Inputs to the ECRIP include:

AUCOST	average unit cost
PTCOST	production total cost
PRCOST	prototype cost (total manufacturing, tooling and test equipment cost of the prototypes)
DTCOST	development total cost (total engineering and manufacturing cost of development phase)

A value for the prototype multiplier (PRMULT) calibration factor is obtained during the calibration process.

4.2.4.4 Complexity of engineering. The experience, skill and know-how of the assigned individuals or team, as applicable to the specified task. This is a measure of the complicating factors of the design effort.

ECMPLX	engineering complexity
SE	systems engineering factor (a function of engineering complexity and development schedule, this factor multiplies the total drafting and design costs to obtain the SE cost)

4.2.4.5 New design and/or design repeat. How much new work is required. The amount of design that can be taken from existing design drawings and the amount of structure repetition.

NEWST	new structure (amount of new structural design effort)
DESRPS	design repeat of structure (amount of structural repetition in a particular design)
NEWEL	new electronics
DESRPE	design repeat of electronics

4.2.4.6 Schedule impact. The relative impact of known and unknown scheduling conditions on the normal time required to complete the project.

PSF	prototype schedule factor
DSTART	development start date
DEND	development end date
DFPRO	development first prototype complete date
DLPRO	development last prototype or completion date
PSTART	production start date
PFAD	production first article delivery date
PEND	production end date
TCALD	time calibration multiplier for development schedule
TCALP	time calibration multiplier for production schedule
NSHIFT	number of work shifts in production phase
NFACS	number of facilities in production phase

4.2.4.7 Technology growth. The technology of hardware production is continually changing. On-going innovations lead to more efficient manufacturing processes, materials, support tools and management practices.

YRTECH	year of technology
ZTECH	technology improvement Z-curve (allows user to control rate of technology improvement)
TECDEL	technology delay (allows forward or backward time adjustment to technology improvement curve)

4.2.4.8 Hardware/software integration. When hardware relies on software for operation, it is necessary to integrate the software with the hardware.

HSINT	hardware/software integration factor
LANG	source language used in the software development effort

SLOC	number of source lines of code excluding comments
FRAC	fraction of non-executable code (DATA statements, etc.)
APPL	application (ranges from simple applications to complex real-time command and control applications)
CPLXM	management complexity (e.g. software developed at more than one location)

4.2.4.9 System integration. Many large hardware developments involve the merging of two or more related hardware products into a single unified system. The individual products often have widely varying characteristics, and they may even be developed by different organizations or companies. Resources and time are required to accomplish total system integration. Cost and schedule estimates are developed for this activity by examining the level of integration required for each individual subsystem, and using the results to determine the effort required to bring subsystems together into a total unified operation.

QTYNHA	quantity next higher assembly (number of units to be integrated and tested at next higher assembly level)
INTEGE	electronic integration factor
INTEGS	structural/mechanical integration factor
EPLANS	electronic plans and procedures as related to integration effort
SPLANS	structural plans and procedures as related to integration effort

4.2.4.10 Specialized costs. Inputs for the 5 specialized elements are readily obtainable and in many cases provided as part of the design effort. Purchased elements use actual costs (including handling), and estimates for given cost elements (e.g., multi-chip modules and custom ICs) are available from PRICE M when actual costs are not available. Costs associated with furnished elements, thru-put elements (items added to the total system cost without any additional markup) and multiple lot production are listed.

COST	recurring cost of purchased items
COSTTYPE	cost type (constant vs. "as spent" units)
CDFRAC	fraction of a custom design cost allocated to a module cost for a given cost element
DDRCST	development drafting cost of a given cost element

DDRAFT	development drafting calibration multiplier
DDECST	development design cost of a given cost element
DDSIGN	development design calibration multiplier
DSYCST	development system cost of a given cost element
DPJCST	development project management cost of a given cost element
DPROJ	development project management calibration multiplier
DDACST	development data cost for a given cost element
DDATA	development data calibration multiplier
DPRCST	prototype manufacturing cost of a given cost element
DTTCST	development tooling and test equipment cost for a given cost element
DTLGTS	development tooling and test equipment cost calibration multiplier
GDTLGT	global development tooling and test sets calibration multiplier (used when DTLGTS is zero)
PDRCST	production drafting cost for a given cost element
PDRAFT	production drafting global multiplier (used to adjust drafting costs without affecting other costs)
PDECST	production design cost of a given cost element
PDSIGN	production design calibration multiplier
PPJCST	production project management cost of a given cost element
PPROJ	production project management calibration multiplier
PDACST	production data cost for a given cost element
PDATA	production data calibration multiplier
PPRCST	production "production" (fabrication, assembly and test) of a given cost element
PTTCST	production tooling and test equipment cost for a given cost element
PTLGTS	production tooling and test equipment cost calibration multiplier
GPTLGT	global production tooling and test sets calibration multiplier (used when PTLGTS is zero)

DCOST	development cost of a thru-put element
PCOST	production cost of a thru-put element
TCOST	total cost of a thru-put element
PIF	PRICE improvement factor (how cost/quantity impacts production) for multiple lot production
UNITLC	unit learning curve for multiple lot production
RATE	production rate in units per month
RATOOL	rate tooling for high production rate multiple lots
GAP	production break (months)
GAPFAC	gap factor to adjust for loss of learning between interrupted multiple lots
LOTFAC	lot factor to adjust for transitions between lots in multiple lot production
OPC	only piece cost (cost of producing only one unit)

4.2.4.11 Other costs. Pertinent escalation rates and mark-ups for general and administrative charges, profit, cost of money, internal research and development, tooling and test equipment cost and cost of engineering change notices.

PTLGTS	production tooling and test equipment
ETLG1	electronic tooling and test equipment multiplier for initial setup
ETLG2	electronic tooling and test equipment multiplier for maintenance costs
STLG1	structural tooling and test equipment multiplier for initial setup
STLG2	structural tooling and test equipment multiplier for maintenance costs
YRBASE	base year economics (inflates actual costs from previous projects for present-day use)
YRECON	year of economics (defines economic base of output costs)
DLEVE	design integration level for electronics (in-house effort required for purchased or furnished items)

DLEVS	design integration level for structure (in-house effort required for purchased or furnished items)
DMULT	development multiplier (linear multiplier to all development cost outputs for markups)
PMULT	production multiplier (linear multiplier to all production cost outputs for markups)
SYSTEM	development systems cost calibration multiplier
ECNE	engineering change notices, electronic (linear multiplier represents percentage of electronic drawing package that will change during production)
ECNS	engineering change notices, structural (linear multiplier represents percentage of structural drawing package that will change during production)

4.2.5 PRICE-HL Life Cycle System/Assembly Control

MTBF	mean time between failures, assuming corrective, not preventative, maintenance.
TF	time to repair LRU
TMO	time to repair module at organization
EE	equipment per equipment location
FN	allowable failure number of LRUs
CEND	cost of engineering department
CPE	cost of production engineering
CUR	contractor unit repair
CMR	contractor module repair
TRE	meantime to repair on-equipment failures
P	number of module types
PP	number of part types
FNSP	fraction of non-standard parts
CPPE	cost of a piece-part replaced on equipment

CFIM	cost of fault isolate to module test equipment
CFIP	cost of fault isolate to part test equipment
FTSQF	foot square floor area for LRU test equipment
FTSQP	foot square area for module test equipment
TC	time to perform ceckout of LRU
CCOU	cost of checkout of LRU support equipment
FTSQC	foot square area for LRU checkout test equipment
DSTART	development start date
DEND	development end date
PSTART	production start date
PEND	production end date
CUP	average cost of a LRU in production
CMP	average cost of a module in production
CPP	average cost of a part in production
YRECON	year of economics
YAT	yearly attrition factor

4.2.6 SEER-SEM software estimation model.

The SEER Software Estimation Model creates cost, schedule, risk, and maintenance estimations for software development. In SEER-SEM, software volume is the primary driver. It can be entered as functions, as lines of code, or as both.

The WBS (Work Breakdown Structure) divides the overall project into computer programs or Computer Software Configuration Items (CSCIs)--the highest unit of a software application--which can be further subdivided into Computer Software Components (CSCs), which can be further subdivided into Computer Software Units (CSUs). SEER-SEM provides cost estimates for each of the following project phases:

1. System concept
2. System requirements design

3. Software requirements analysis
4. Preliminary design
5. Detailed design
6. Code and CSU test
7. CSC integrate and test
8. CSCI test
9. System integrate through operational test and evaluation
10. Maintenance and operation support.

These phases correspond to the traditional waterfall model of development which may not apply to the RASSP design methodology, but is appropriate for representing current practice.

Built-in knowledge bases are chosen as a function of four characteristics--platform (avionics, business, ground, manned space, missile, mobile, ship, unmanned space), application (CAD, command/control, data base, diagnostics, flight, message switching, MIS, mission planning, MMI/graphics, office automation, OS/executive, process control, radar, report generation, simulation, software development tools, test, training, utilities, other), development method (Ada development, Ada development with incremental methods, Ada full use, prototype, spiral, traditional incremental, traditional waterfall), and development standard (commercial, 2167A, 2167, 2167A minimal set, 2167A full set, 1703, 483-490, 1679 with IV&V.)

The values of the aforementioned four characteristics define a specific type of WBS item which SEER-SEM uses to generate the most likely values and ranges for an extensive list of input parameters. These parameters can then be modified by the user to further customize and refine the model of the overall project environment. The parameters are divided into sixteen categories:

4.2.6.1 Effective Size. Includes the following parameters:

4.2.6.1.1 New Lines of Code.

4.2.6.1.2 Pre-Exists, Not Designed for Reuse.

- Pre-Existing Lines of Code
- Lines to be Deleted in Pre-Existing
- Lines to be Changed in Pre-Existing
- Percent to be Redesigned
- Percent to be Reimplemented
- Percentage to be Retested

4.2.6.1.3 Pre-Exists, Designed for Reuse.

- Pre-Existing Lines of Code
- Lines to be Deleted in Pre-Existing
- Percentage to be Redesigned
- Percentage to be Reimplemented
- Percentage to be Retested

4.2.6.2 Complexity. An overall rating of the software's inherent difficulty.

4.2.6.3 Personnel Capabilities & Experience. Includes the following parameters:

- analyst capabilities
- analyst application experience
- programmer capabilities
- programmer language experience
- development system experience
- target system experience
- practices & methods experience

4.2.6.4 Development Support Environment. Includes:

- modern development practices use
- automated tools use
- logon through hardcopy turnaround time
- terminal response time
- multiple site development
- resource dedication
- resource and support location
- host development system volatility
- practices and methods volatility

4.2.6.5 Product Development Requirements. Includes:

- requirements volatility
- specification level/reliability

- test level (verification/validation)
- quality assurance level
- rehost from development to target

4.2.6.6 Product Reusability Requirements. Includes:

- reusability level required
- software impacted by reuse

4.2.6.7 Development Environment Complexity. Includes:

- language type complexity
- host development system complexity
- application class complexity
- practices and procedures complexity.

4.2.6.8 Target Environment. Includes:

- special display requirements
- memory constraints
- time constraints, real time code
- target system complexity
- target system volatility
- security requirements

4.2.6.9 Schedule (optional). Includes the required schedule (in calendar months)

4.2.6.10 Staffing (optional). Includes:

- maximum staffing rate per year
- maximum total staff available
- maximum effort available (in man-months)

4.2.6.11 Probability. An overall probability of completion for the software job under estimation.

4.2.6.12 Software Requirements Analysis. Includes:

- requirements complete at contract

- requirements definition formality
- requirements effort after baseline

4.2.6.13 Software to Software Integration. Includes:

- CSCIs concurrently integrating
- integration organizations involved
- external interfaces among CSCIs.

4.2.6.14 Software to Hardware Integration. Includes:

- hardware integration level
- unique hardware interfaces.

4.2.6.15 Software Maintenance. Includes:

- years of maintenance
- separate sites
- maintenance growth over life
- personnel differences
- development environment differences
- annual change rate
- maintain total system.

4.2.6.16 Other Add-ons. Includes:

- external QA Costs
- program office costs
- IV&V costs.

4.2.6.17 Average personnel costs. The average costs per labor-month for the base year which consists of:

- direct software management
- software system requirements analysis
- software requirements analysis
- software design

- software programming
- software quality assurance
- software configuration management
- software data preparation.

4.2.7 SEER-SSM software sizing model.

The SEER software sizing model estimates the expected size of a software project based on qualitative/relative inputs without the use of databases.

As in SEER-SEM, the WBS (Work Breakdown Structure) partitions the overall project into modules--CSCIs which can be further divided into CSCs which can be further divided into CSUs--whose operational and functional characteristics are defined. SEER-SSM customizes the requirements for user-provided input after the partitioned modules to the model have been designated.

SEER-SSM requires project information (company/organization, project name, file name), module data (name of software unit and at least two reference modules of known size with their size expressed as in DSI, DEMI, or function point count), and four user-provided input data sets (DSXs)--pairwise data, PERT sizing data, sorting data, and ranking data--for execution.

4.2.7.1 Pairwise Data. SEER-SSM provides unique random pairings of all modules in the project and requires the user to make a binary decision concerning their comparative sizes.

4.2.7.2 PERT Sizing Data. SEER-SSM requires the user to estimate:

- the total number of lines of code providing the lowest possible size for each module
- the most likely size for each module
- the highest possible size for each module

4.2.7.3 Sorting Data. SEER-SSM provides a number of size intervals and the user is to determine in which interval the size of each particular module falls.

4.2.7.4 Ranking Data. SEER-SSM provides unique ordered pairings (ordered tentatively after the three previous steps) of modules in the project and requires the user to make a binary decision concerning their comparative sizes.

4.2.8 SEER-IC integrated circuit model.

SEER-IC uses a Work Breakdown Structure (WBS) to create cost estimates for integrated circuits (chips), multi-chip modules (MCMs) and chips on MCMs. Built-in and customized knowledge bases may be used to provide information for estimates. Built-in knowledge bases are selected as a function of project type (MCM, complex gate array, custom chip, monolithic microwave integrated circuit, "none," semi-custom chip or simple gate array), platform standard (industrial, commercial, military airborne, military ground, military ground mobile, military sea, "none," manned space or unmanned space) and acquisition category (buy and integrate, customer furnished equipment, make, "none," or subcontracted item). User created (class) knowledge bases can be created if desired. Adjustment factors can be applied for specification generation, design, prototype hardware and average unit production in each of the class, platform standard and acquisition category knowledge bases. Such adjustments are used to accommodate variations due to fees or discounts. Once the applicable knowledge bases have been invoked and adjustments applied, information is entered to perform estimates. Most input variables have an optional associated range such as "least, likely, most," or "low, nominal, high." Application ranges for all required inputs (except production quantity) are loaded by the knowledge bases. Users narrow the input ranges when actual values are known. Inputs required to perform an estimate fall into 10 categories as described in the following.

4.2.8.1 Product description. Includes:

- Chip area (die area in square millimeters)
- MCM substrate area
- number of devices on MCM
- feature size (minimum line width and spacing in microns)
- transistors per chip
- gates per chip
- input/output pins per chip or MCM (including power)
- process type (wafer technology or material used such as CMOS exotic material, GaAs, linear, NMOS, PMOS, SOS or TTL)
- package type (DIP, flatpack, leadless chip carrier, pin grid array or unpackaged die)
- wafer diameter and operating frequency (very high for >500 MHz, high for 200-500 MHz, nominal for 50-200 MHz, low for 15-50 MHz and very low for <15 MHz).

4.2.8.2 Mission description. Includes:

- Chip classification (custom, semi-custom, complex gate array or simple gate array)
- operating environment (commercial, military or space).

4.2.8.3 Program description. Includes:

- New design (specifies percentage of design which is new)
- iterations (number of re-design and re-manufacture cycles to be done on prototype units until satisfactory performance is obtained)

- certification level (very high for class S, high for upscreened class B, nominal for class B, low for industrial and very low for commercial grade devices).

4.2.8.4 Development environment. Includes:

- Developer capability and experience (very high for an experienced team in the 90th percentile, high for 75th percentile, nominal for 50th percentile, low for 30th percentile and very low for a novice team in the 5th percentile)
- development tools and practices (very high for an organization with modern development practices and tools in the 90th percentile, high for 75th percentile, nominal for 50th percentile, low for 30th percentile and very low for an organization in the 5th percentile using only stand-alone tools with no logic/timing/fault simulation)
- requirements volatility (extra high for frequent moderate and major changes, very high for frequent moderate and occasional major changes, high for occasional moderate changes, nominal for occasional minor changes and low for essentially no requirements changes).

4.2.8.5 Production environment. Includes:

- Production experience (very high for a near-perfect 90th percentile production team, high for 75th percentile, nominal for 55th percentile, low for 35th percentile and very low for a non-functional team in the 5th percentile)
- production tools and practices (very high for a fully automated large scale facility less than 2 years old, high for a fully automated large-to-medium scale facility, nominal for highly automated medium scale facility, low for a semi-manual small scale facility and very low for a prototyping facility).

4.2.8.6 Program schedule. Includes:

- Start date for development
- prototype quantity
- start date for production
- optional specified yield (percentage of production units surviving testing operations).

4.2.8.7 Production. Includes:

- Prior production units (number of units previously produced that should be credited to this program)
- total production quantity
- percentage of item purchased (percentage of item that will be developed elsewhere and integrated into the system as a purchased item or customer-furnished equipment)

• production unit purchase cost (thruput of costs for purchased items not included in the WBS).

4.2.8.8 Probability. Includes the probability that the estimate will not exceed actual cost (90% used in risk analysis for worst-case estimate, 80% for fixed price bids, 50% for nominal and 20% for cost plus development).

4.2.8.9 Economic factors. Includes:

- Development fee (percent of development costs to be added to the estimate to account for additional fees)
- production fee (percent of production costs to be added to the estimate to account for additional fees).

4.2.8.10 Project parameters. Includes:

- System quantity (the number of systems being built)
- fiscal year start month
- currency exchange rate
- base year (the year which represents the base of the constant-year dollars)
- cost escalation factor (inflation/deflation factor to convert base year dollars to then-year dollars)
- database (e.g., the seeric93 database is chosen for performing estimates with 1993 technology)

4.2.9 SEER-H hardware estimation model.

SEER-H uses a Work Breakdown Structure (WBS) to create cost estimates for hardware elements. Built-in and customized knowledge bases may be used to provide information for estimates. Built-in knowledge bases are selected as a function of element type (mechanical or electronic), application (hydraulics, signal processor, communications, etc.), platform (ground, air, space, fixed or mobile, manned or unmanned), development standard (commercial, military specification), and acquisition category (buy and integrate, customer furnished equipment, make, subcontracted, or "none"). User created knowledge bases (class) can be created if desired. Adjustment factors can be applied for specific generation, design, prototype hardware, and average unit production in each of the class, platform standard, and acquisition category knowledge bases. Such adjustments are used to accommodate variations due to fees or discounts. Once the applicable knowledge bases have been invoked and adjustments applied, information is entered to perform estimates. Most input variables have an optional associated range such as "least, likely, most," or "low, nominal, high." Inputs required to perform an estimate fall into 11 categories as described in the following.

4.2.9.1 Inputs unique to electronic WBS elements.

4.2.9.1.1 Product description. Includes:

- Total number of printed circuit boards (PCBs)
- Circuitry composition (analog, digital, hybrid, optical)
- discrete components per PCB
- integrated circuits per PCB
- I/O pins per PCB
- clock speed
- packaging density (extra high for all MCMs, very high for many MCMs, high for some MCMs but mostly individual packaging, and nominal for no MCMs)
- IC technology (very high for ULSI, high for SLSI, nominal for VLSI, low for LSI, very low for MSI, and very low- for SSI).

4.2.9.1.2 Mission description. Includes:

- Operating environment (air, ground, sea, and space)
- electronics classification (comm, comp, C/D, electromagnetic, nav)
- electronics fault detection
- electronics fault isolation.

4.2.9.1.3 Program description. Includes:

- New design (percentage of design that is new)
- design replication (percentage of design that is not unique)
- certification level (very high for manned space product, high for unmanned space product, nominal+ for military aircraft product, nominal for commercial aircraft product, nominal- for military ground-mobile or sea product, low for military ground system, and very low for commercial grade)
- hardware integration level (very high for 3-4 levels of integration, high for 2-3 levels of integration, nominal for 1-2 levels of integration, low for 1 level of integration, and very low for no integration requirements).

4.2.9.2 Inputs unique to mechanical WBS elements.

4.2.9.2.1 Product description. Includes:

- Weight (pounds or kilograms)
- volume (cubic feet or liters)
- material composition (percent aluminum/malleable, steel alloy, commercial exotic, other exotic, composite, polymer, ceramic)
- complexity of form (extra high for highest precision level for assembly, very high for precision assembly, high for assembly but no internal movements, nominal for assembly with multiple fasteners but no internal movements, low for simple assembly with standard fasteners, and very low for no assembly)
- complexity of fit (extra high for tolerances less than 1.5 mils, very high for tolerances between 1.5 and 5 mils, high for tolerances between 5 and 10 mils, nominal for tolerances between 10 and 20 mils, low for tolerances between 20 and 40 mils, and very low for tolerances between 40 and 60 mils)
- construction process (very high for highly labor intensive fabrication and assembly, high for moderately labor intensive fabrication and assembly, nominal for low labor intensive fabrication and assembly, low for minimum labor intensity operations with 50% robotic assembly, and very low for single operator, 100% robotic assembly).

4.2.9.2.2 Mission description. Includes:

- Operating environment
- hardware classification (structure, mechanical, hydraulic/pneumatic)
- operating service life (in hours)
- internal pressure (in psi or kN/m2, 8000 psi is very high and 700 psi is very low).

4.2.9.2.3 Program description. Includes:

- New design
- design replication
- certification level
- hardware integration level.

4.2.9.3 Inputs common to both electronic and mechanical WBS elements.

4.2.9.3.1 Development environment. Includes:

- Developer capability and experience (very high for 90th percentile, high for 75th percentile, nominal for 55th percentile, low for 35th percentile, and very low for 5th percentile)
- development tools and practices (very high for CAD/CAM, high for automated tools, nominal for use of but no experience in CAD but not CAM, low for experimental use with automated tools, and very low for no use of automated design or manufacturing)
- requirements volatility (very high for frequent moderate and major changes, high for frequent moderate and occasional major changes, high for occasional moderate changes, low for occasional minor changes, and low for essentially no changes at all).

4.2.9.3.2 Production environment. Includes:

- Production experience (very high for 90th percentile, high for 75th percentile, nominal for 55th percentile, low for 35th percentile, and very low for 5th percentile)
- production tools and practices (very high for CAD/CAM, high for automated tools, nominal for use of but no experience in CAD but not CAM, low for experimental use with automated tools, and very low for no use of automated design or manufacturing).

4.2.9.3.3 Program schedule. Includes:

- Required development schedule
- development start date
- prototype quantity
- production start date
- production learning curve (Cumulative Average, Unit Theory)
- prior production units
- production quantity.

4.2.9.3.4 Purchased items. Includes:

- Percentage of item purchased
- production unit purchase cost
- unit purchase cost
- probability (90% usually worst case estimate, 80% fixed price bid, 50% most likely, 20% optimistic).

4.2.9.3.5 Economic factors. Includes:

- Engineering hourly rate
- manufacturing hourly rate
- material cost per PCB/pound.

4.2.10 SEER-HLC hardware life cycle model.

4.2.10.1 Project level parameters. Includes:

- Project name
- Operations & Support start date
- O&S duration
- inflation rate
- fiscal year start month
- cost input base year
- organizational alternate repair hourly labor rate
- intermediate alternate repair hourly labor rate
- depot alternate repair hourly labor rate.

4.2.10.2 Site parameters. Includes:

- Site identifier
- maintenance shifts/week
- system quantity
- date operations begin
- date operations end.

4.2.10.3 Common support equipment parameters. Includes:

- Support suite
- production cost/suite
- date available.

4.2.10.4 Prime mission equipment parameters. Includes:

- WBS

- PME equipment name
- quantity per system
- shipping weight
- operating hours per month
- PME operating hours to maturity
- PME replacement cost
- spares sufficiency probability
- consumable cost per repair
- annual resource cost
- PME mature mean time between failure
- condemnation rate
- retest OK rate.

4.2.10.4.1 PME item organizational maintenance details. Includes:

- Mean time to repair at organizational level
- in place repair rate
- organizational shared support equipment
- organizational hourly labor rate
- organizational peculiar support equipment date available
- organizational PSE unit cost.

4.2.10.4.2 PME item intermediate maintenance details. Includes:

- Mean time to repair at intermediate level
- intermediate turn around time
- intermediate shared support equipment
- intermediate hourly labor rate
- intermediate PSE date available
- intermediate PSE production unit cost
- Not Repairable This Station rate.

4.2.10.4.3 PME item depot maintenance details. Includes:

- Mean time to repair at depot level
- depot turn around time
- depot shared support equipment
- depot hourly labor rate
- depot PSE date available
- depot PSE production unit cost.

4.3 Process and Product Metrics

The metrics identified in Section 4.2 are extracted from commercially available packages for program management. These metrics attempt to quantify factors that effect the (generalized) cost of a project such that each package gives a comprehensive picture of the development and production costs of a project.

The metrics in Section 4.3 are directed toward specific issues of performance of both the RASSP process and products, and complexity of the products. For the most part, metrics for the complexity of the RASSP process, such as the total number of source lines of code in the RDE, are not required. The complexity of the RASSP process is measured indirectly through productivity metrics, cost of the tools, and ease of use.

Section 4.3.1 is concerned with a evaluation of the RASSP *process* as it is applied to develop a product, and not an evaluation of the end *product*. However, process performance is not unrelated to the products being developed, and so measures of product complexity and performance are required to fully comprehend the process performance. Without measures of product complexity, the performance of the RASSP process for different products or benchmarks cannot be compared. Without an understanding of product performance, success of the RASSP process cannot be quantified or compared to current practice. Products include not only the final hardware and software constituting the embedded signal processor, but also a host of intermediate and supporting items such as documentation, simulation models, schedules, and life cycle cost estimates. Comprehensive and detailed metrics cannot be collected for every intermediate RASSP product on every benchmark.

Some of the metrics are subjective in nature, while others are specific. The more subjective metrics are apt to be found in all phases of the project, while specific quantitative metrics are usually limited in their focus. Metrics are intended to measure the performance of the process and not the people, although the expertise of the people will influence the success and efficiency of the process. Metrics are not intended to be used to analyze the performance of individuals involved in the benchmark execution.

4.3.1 Design process

The different tools and procedures that are used in all segments of the benchmark execution are considered in the evaluation. Metrics must be collected to quantify the value of both the tools and the underlying methodology. Although the ultimate measure of success is the reduction in the design cycle time, analysis of progress during the RASSP program requires an understanding of which steps in the RASSP process consume the majority of the time, and where improvements in the time required to execute the process are occurring.

4.3.1.1 Tool Evaluation. For each major tool used during execution of the benchmark, the metrics indicated in Table 9 are required. The time spent using the tool can be expressed as a relative or percent

Table 9. Tool Evaluation Metrics

Measurement	**Description**
Time	Time usage associated with each tool (TOOL_USAGE)
Assessment	Value of tool to the process (TOOL_VALUE)
Assessment	Heterogeneous platform support (TOOL_OPEN)
Assessment	Seamless access to design data (TOOL_DACCESS)
Assessment	Human interface (TOOL_GUI)
Assessment	Interoperability (TOOL_INTFCE)
Assessment	Unified project data management (TOOL_PROJDAT)
Assessment	Consistent process management (TOOL_PROJMGT)
Assessment	Comprehensive library management (TOOL_LIBMGT)

time for the overall process or step in the process. The assessments in Table 9 will be assigned a quality factor of from zero to four, with zero corresponding to the poorest assessment and four the best. The value of the tool ranges from nonessential, meaning the step could be completed via a variety of other tools or methods, to essential, meaning that the tool is critical to a timely execution of the step.

4.3.1.2 Complexity. The size of a project contributes to complexity in a non-linear manner. Something that can be visualized by a single individual is not linearly scalable to a large project with its attendant interfaces. Well-defined interfaces using standard protocols or previously developed interfaces are preferable to custom designs. A project must be divided into subprojects which are distinct with clean interfaces. The number of disciplines involved in a project contributes to complexity. A project in which the systems engineer can reasonably be expected to be trained is inherently less complex than another in which that person must rely on the technical and managerial advice of others. Solutions which push the state of the art will contribute to complexity. The benefits of repeated utilization are not realized for the first implementation. Three fundamental elements will form the basis of the complexity measurements.

These are reuse, interfaces, and comprehension.

4.3.1.2.1 Reuse. The services in this category must perform as advertised and as might be expected by a reasonable person. Elements in the reuse library which do not perform as advertised will impact the complexity because they distort the planning process and the schedule. A trigonometric function in a computer program is not inherently more complex than a multiply instruction because it is part of the library. The maturity of the reuse library and the experience of the users are important.

The metrics required to evaluate reuse relate to the time saved through use of this technique. This requires meaningful estimates of the time that would have been spent in creating an original design, the time spent in exploring and evaluating the elements of the reuse library, the time spent in incorporating elements of the reuse library into the applicable design, and finally the time spent in re-evaluating the decision because the description of the reuse elements was inadequate or faulty (a defect, report as specified in Section 4.3.1.3). By using elements of a library, fewer defects are introduced into the design and this must be estimated from the anticipated defect rate. Because these quantities are sometimes nebulous, it shall also be required that the Developer estimate the time and cost saved in bringing the product to the end user (REUSE_ENT_T and REUSE_END_C). In the case of software, the reuse metric shall also be expressed as a percentage of the executable lines of code. This, however, does not by itself measure the complexity of the code from the reuse library. This shall be estimated from the time saved as discussed earlier in this paragraph. The specific tool associated measurements and metrics in both time (person-hours) and cost are listed in Table 10.

Table 10. Reuse Measurements and Metrics

Element	**Description**
Original Implementation (time or cost)	Estimated (time or cost) for original design and implementation (REUSE_ORIG_T, REUSE_ORIG_C)
Reuse evaluation (time or cost)	Time (or cost) expended in learning the capabilities of the reuse library (REUSE_EVAL_T, REUSE_EVAL_C)
Reuse Incorporation (time or cost)	Time (or cost) actually spent incorporating element of reuse library including implementation* (REUSE_T, REUSE_C)
Reuse Time Metric improvement	original implementation time / reuse incorporation time (REUSE_TRATIO)
Tool Cost Metric improvement	original implementation cost / reuse incorporation cost (REUSE_CRATIO)

* The implementation must allow for the fewer defects that would be generated by reuse as compared with an original design.

The specific software associated measurements and metrics for VHDL and Ada (or other high level language) are listed in Table 11. FPGA "software" is included in this category.

Table 11. Software Reuse Measurements and Metrics

Element	Description
Original (time or cost)	Estimated (time or cost) for original design and implementation (S_REUSE_ORIG_T, S_REUSE_ORIG_C)
Reuse evaluation (time or cost)	Time (or cost) expended in learning the capabilities of the reuse library (S_REUSE_EVAL_T, S_REUSE_EVAL_C)
Reuse (time or cost)	Time (or cost) actually spent incorporating element of reuse library including implementation* (S_REUSE_T, S_REUSE_C)
Software Time Metric improvement	original implementation time / reuse incorporation time (S_REUSE_TI)
Software Cost Metric improvement	original implementation time / reuse incorporation time (S_REUSE_CI)
Reuse code (percent)	NCSS saved through reuse / NCSS including reuse library (S_REUSE_LOC)
Defect Density	Estimated defects per 1K NCSS (S_REUSE_DEFECT)

* The implementation must allow for the fewer defects that would be generated by reuse as compared with an original design.

The experience of the people on a project constitutes an element of the reuse concept. That experience does not alter the complexity of the hardware or software which is being implemented but it does have an impact on the process. Therefore it does enter the equation for describing the complexity of the RASSP implementation of the system. A breadth of experience is in the same category. It is here that a more global understanding of the project goals can supply the feedback that is so important to co-design which may then alter the distribution of resources or complexity in a more optimal manner. Metrics applicable to people have been described in Section 4.2.1.4, Section 4.2.4.4, and Section 4.2.6.3.

4.3.1.2.2 Interfaces. These are, classically, an area which contributes significantly to complexity. Whether it be the acquisition of data through custom hardware interfaces or the control and coordination of other functions, these interfaces and the disparity of elements passing through them contribute to complexity. The reuse function invoked by the use of standards will always be beneficial where they are appropriate. This is not everywhere. It is possible to misuse standards through a lack of understanding of the underlying protocols. Because good software design breaks down a complicated problem into a large number of small modules, there are a large number of interfaces to contribute to complexity. The type of data being transmitted must be the same as the type being received. This may seem obvious from a hardware point of view but not all software languages have supported this important feature in the past. An interface is between two bodies; it should not alter the data on a different interface somewhere else in the system. Again, this may seem obvious from the hardware point of view but it has not always been possible for programmers to encapsulate the data passed between tasks. Modern languages and good software engineering practices contribute to a minimization of complexity. Interfaces should have a handshake. To ignore a handshake when it is offered risks being oblivious to error messages returning from other modules. Specific metrics for VHDL and Ada software are described in Section 4.3.4.7.

4.3.1.2.3 Comprehension. Comprehension recognizes the limits of a human being to absorb the complexity of a problem. It is for this very reason that large problems are successively divided up into pieces such that the smallest piece can be fully comprehended by a single person at any point in time. Even in small teams, each individual has an immediate task and a set of interfaces to the team members. This task may change dynamically with the team members over time but never should be left so unwieldy that it is incomprehensible. In software it has long been recognized that control is more complex than sequential processing. The number of "if, while, and for" statements in a module is an important measure of complexity. The "goto" statement has long been out of favor because of its supreme ability to obfuscate a simple program. Metrics applicable to this arena are described in Section 4.3.4.4.The exclusive use of upper case is equally damaging to comprehension. For over a thousand years our language has made use of upper and lower case to enhance our ability to fathom the written page, only to have been destroyed by the introduction of Fortran in 1957. The metrics described in Section 4.3.4.2 will also be used to evaluate comprehension. Modern programming practices and good practice foster comprehension.

The first generation of commercial array processors were exceptionally obtuse in the software department, sometimes requiring as many as three unique languages for all portions of the processor. Just as hardware has improved and matured over the years, so too has the software. Today, a modern array processor can be programmed in a high level language with a compiler. For improved efficiency, vendors supply libraries of functions which may have been hand coded in assembly language or worse. Writing microcode for a pipelined architecture stresses the limits of complexity. Metrics applicable to microcode are described in Section 4.3.4.8.

4.3.1.3 Defects. Defects are defined from a customers point of view. A deliverable to a customer that is not within specifications is considered to be a defect. The customer may, in the tradition of Total Quality Management, be internal. The customer may be in the next office; the customer may be the same team, working on the next stage of the product. Defects that occur during development, prior to delivery, are considered private and will not be counted as defects. A hardware failure, while important from the point of view of reliability, is not a RASSP *process* defect, unless it can be shown to have arisen from a design flaw such as inadequate cooling. Within the spirit of this definition, a declared limited functionality is not a defect. But a delivery to a customer must function correctly within the declared scope of that delivery. This definition is in concert with the spiral model. A change in requirements is not a defect. Defects are not simple black or white objects, they are complex elements which must be understood in the overall context of the project. Defects implicitly receive weighting factors which are a function of the time and space in which the defect seems to be located. Defects which are caught early in the process have little weight, while those not caught until equipment is in the field have a ponderous weight. Defects which have a small sphere of influence also have little weight, while those which have ramifications far removed from their own location are weighted more. The time in the benchmark cycle when a defect is identified (DEFECT_FIND) shall be reported with the Developer's measurement or estimate of the time (person-hours) required for correction. The fundamental source of the defect shall also be reported (DEFECT_SRC). The measurements and metrics for tool defects shall be reported as described in Table 12.

Table 12. Tool Defect Measurements and Metric

Element	Description
Fix time (and cost)	Time (and cost) consumed in fixing defect* after existence was recognized (DEFECT_UNDO_T, DEFECT_UNDO_C)
Lost time (and cost)	Estimated time lost because defect existed (DEFECT_LOST_T, DEFECT_LOST_C)
Lost capability metric (percent)	Increase in "time to market" or IOC as a percent of bench-mark life span (DEFECT_TTM)

* A defect is not simply a bug in the program, an invalid modeling of a simulation is also a defect.

4.3.1.4 Requirements Traceability. Requirements impact the design which in turn impacts the implementation. It is important to know that a design element responds to some requirement and that every requirement has been addressed by at least one design element. Testing may produce changes in any of these elements and the relationships between the requirements and the design elements must still exist. There are five general areas in the implementation of the requirements. These are: requirements, design, implementation, tests, and changes. The implementation may be hardware or software, whether the latter be VHDL or Ada or C code. A partial list, as an example only, of the document types that are produced is given in Table 13.

Table 13. Partial List of Document Types

Document Type
Requirements Specification
Software Design Document
VHDL Design Language File
Source Code
VHDL Test Description
VHDL Test Results
User's Manual
Project Development Schedule
Software Problem Report Database

Documents normally contain data only of the same kind, e.g., the requirements specification document contains the requirements data for the implementation. This packaging is quite natural since similar data are normally created together at the same time by the development team. However, this packaging scheme fails to capture important relationships between engineering data of different kinds. This scheme will not assist the development team in tracing how any particular requirement is allocated to specific design elements or how these design elements are implemented by the VHDL or Ada code elements. The metric associated with requirements traceability for the documentation associated with each tool shall be calculated based on Table 14.

Table 14. Requirements Traceability Metric (REQ_TRAC)

Level	Characteristic
0	Completely manual process; documents not under a version control system; no links between documents
1	TBD
2	Some documents under version control; some links may exists between documents in different areas
3	TBD
4	Full hypertext documentation with anchors and links between all five areas. Interactive navigation of the linked elements. Complete backannotation between tools, support for concurrent processes. Full configuration control

4.3.1.5 Measurements. At the start, the most important element in calculating metrics will be an acceptance of the need for the measurements. This is part of the total quality management. Without a commitment on all levels, it cannot be successful. It must be recognized and accepted by management that the collection of measurements is an important part of this program. It must be accepted by, and passed down from, management. Measurements are not collected at the end of the project, they are part of a continuous process which enables trend data to be made available.

4.3.2 Application Complexity Metrics

Application complexity metrics focus principally on the products or applications developed with RASSP, and are viewed as a means of normalizing different benchmarks so that comparisons of the RASSP process over time and over different applications can be made.

Application requirement complexity metrics (ARCMs) endeavor to capture the inherent complexity of a given benchmark application, independent of the particular hardware and software implementation. The ARCMs form the basis for comparing the difficulty of successive benchmarks. The ARCMs will also serve as a reference for determining efficiency of the hardware and software realizations of a benchmark produced by the Developer.

ARCMs consist of three components: application requirements, external constraints, and "ility" requirements. The following ARCMs shall be computed by the Developer with reference to the processor requirements of Section 2 and the executable requirements of Section 3.

4.3.2.1 Application requirements. The complexity of any embedded signal processor is determined by the inherent complexity of the application being implemented. The complexity of the application is determined by its function, computational requirements, control flow, external interfaces, and dynamic range or precision.

To determine functional complexity, the total number of system operations required per output datum shall be recorded (TOTSYSOP). A system operation is any uniquely defined mathematical operation on one or two input variables that generates a single output. Straightforward mappings of single variables, such as trigonometric functions, logarithms, and exponentials are considered systems operations. The Fourier transform is a mapping and, thus, is also a system operator. Well-defined algebraic operators between two variables including inner products, vector multiplies, dot-products, and cross-products are also considered system operators.

Convolution is a moving inner-product and is therefore more complex than a system operator. However, convolution is based on a unique system operator (i.e., the inner-product) applied in a series of similar operations. Such re-use of system operations reduces the complexity of the application process. Thus, the number of unique system operators per output datum shall be recorded (UNISYSOP). Operators are considered to be re-used if they act on similar data. For instance, equalization weighting of data of different polarization are similar, but equalization weighting is not similar to kernel multiplication in azimuth compression (see Section 2.1). The maximum number of system operations per unit time (SYSOPS) shall also be recorded.

Computational requirement is a measure of the computation required per second for arithmetic and logical operations. For arithmetic operations both primitive and composite operations are defined. Primitive operations are multiply, add, subtract, compare, shift, etc. and no distinction is made between integer and real operations or single and double precision. If there are a significant number of complicated primitives, such as divide, they should be accounted for as an equivalent number of primitive operations. The PROPS metric is given in primitive operations per second. It is understood that this metric is dependent on the assumed implementation of the algorithm and is meaningful only with an explanation of the assumed implementation. A composite operation is an operation on one or more sets of data which produces another set, such as a Fourier transform, vector multiply, etc. UNICOMOP is the number of unique composite operations in the assumed implementation and COMOPS is the number of primitive operations per second which are accounted for by composite operations. (COMOPS will be a subset of PROPS.)

Control flow (CONFLOW) complexity is a measure of the number of user commanded modes of operation and degree of data dependent branching. It is rated Low, Medium or High.

The maximum number of system data, including constants, coefficients, etc., required to be resident within the application process at any time shall be recorded (SYSRES). Datum types (e.g., frequency samples, range pulses, images,.) vary within the application process, so that SYSRES is a count of mixed datum types. In addition, DATARES is the maximum amount of data required to be resident in the process as measured in bytes. No distinctions are made as to where the data may reside in a possible storage hierarchy.

To determine the complexity of external interfaces, the total number of external interfaces shall be recorded (TOTEXTINT), together with the number and type of unique (UNIEXTINT) and non-standard interfaces (NSTDEXTINT). Average and peak data rates shall be recorded for all process input and output data (AVGIN, PKIN, AVGOUT, PKOUT). The number of input sources (INSOU) and output (OUTDES) destination of process data shall be recorded together with the maximum allowable response latencies (LATENT).

The required dynamic range (DYNAMIC) and precision (PRECIS) of both processor input and output data shall also be recorded.

4.3.2.2 External constraints. External constraints affects the complexity of embedded signal processors. Such constraints include the physical constraints imposed by the system into which the processor is imbedded, as well as environmental and cost constraints and imposed mil-standards.

The physical constraints of the processor shall be recorded. This shall include maximum allowable size (MAXSIZE) and weight (MAXWGT), maximum allowable values of peak and average power (MAXPKPOW, MAXAVGPOW), and the source of prime power (PRMPOW); e.g., 110VAC, 28VDC, etc.

The environmental constraints of the processor shall be recorded. This shall include the allowable ranges for temperature, humidity, altitude, corrosion resistance, and shock and vibration (TEMP, HUMID, ALT, CORRES, SHOCK). Allowable values of all constraints for both operational use and storage shall be recorded.

All cost constraints shall be recorded. This includes total cost (TOTCOST) as well as non-recurring engineering costs (NRECOST).

All required mil-standards shall be recorded as well as any modifications, tailoring, or exemptions to required standards.

4.3.2.3 Ility requirements. Requirements for testability, reliability, and maintainability increase the complexity of the embedded signal processor. The required degree of fault coverage shall be recorded (FLTCOV) together with the maximum allowable latency in detecting faults (FLTLAT) and the required level of fault isolation (FLTISO). The maximum allowable fault rate (MAXFLTRT) shall be recorded together with the maximum allowable time for fault recovery (MAXFLTREC). The skill level required for maintenance personnel shall be recorded (SKILL) as will the required level of documentation (DOC).

4.3.3 Hardware Products

The primary concern of the hardware metrics is the RASSP product. Hardware performance metrics measure the performance of the processor and they mirror the application process requirements metrics of Section 4.3.2.1. Hardware complexity metrics measure characteristics of the hardware implementation. Although Benchmark 1 is concerned with processor prototyping and involves no hardware fabrication estimates may be made of some of these metrics.

4.3.3.1 Hardware Performance Metrics. Wherever possible, in Benchmark 1, the Developers shall provide estimates for the following metrics.

4.3.3.1.1 Execution rate. The execution rate realized in primitive operations per second (as defined in Section 4.3.2.1) while executing the Benchmark application shall be reported as PROPS. If the processor can operate in several different modes then PROPS shall be reported for the each mode. If the entire processor can support a higher throughput than required by the application then the maximum possible rate shall be reported as MPROPS. The subsystem or interface which determines this maximum shall be described.

4.3.3.1.2 I/O and dynamic range. The peak and sustained data transfer rate and the data representation format at each system interface shall be reported. The dynamic range supported in the processing circuits shall be reported.

4.3.3.1.3 Power. Peak power (PKPOW) and average power (AVGPOW) when operating at the rate for which PROPS is reported.

4.3.3.1.4 Size and weight. The dimensions of the system box(es) and their weight.

4.3.3.1.5 Cost. Both real costs of the prototype and projected manufacturing costs are desired. Prototype costs shall include the total small-quantity cost of all components in the system and NRE incurred. The estimate of production cost for producing N units over a period of Y years shall include component, NRE, manufacturing, testing and documentation costs. Unit Life Cycle cost for an assumed total number of N units over a period of LC years shall be reported.

4.3.3.1.6 Testability. The level of conformance to testability specifications as well as any additional capability added by the developer shall be described. The data may represent both estimates and results of experiments and shall include: time to execute routine diagnostics, test coverage, level of fault isolation and mean time to detect faults.

4.3.3.1.7 Reliability/Availability. The level of adherence to reliability/availability specifications shall be described. Data to be presented includes predicted mean time to failure and time to recover from or repair a fault.

4.3.3.1.8 Environment. For both operational and storage environments the design goals and measurement results for temperature, altitude, humidity, and shock and vibration resistance shall be reported.

4.3.3.2 Hardware Complexity Metrics. Hardware Complexity Metrics (HCM) capture the complexity of the benchmark application hardware through measures of degree of integration, COTS vs. custom, number of elements, clock rate, etc. They also give a measure of the level of technology employed.

- Size Storage: For each of the storage levels: cache (off processor-chip), main and second level, report the total number of bytes of storage.
- Gates: For the sum of all ICs comprising COTS MSI devices, FPGAs and custom ICs give the total number of gates in some well defined manner (e.g. and-gate equivalents).
- VLSI: Identify and report number used of all non-memory ICs not included in above gate count, i.e. processors and other function-specific circuits.
- Technology Speed: Report the clock rate of the system and identify any asynchronous circuits and interfaces. Identify any circuits which use higher-speed internal clocks.
- IC: Identify all circuit technologies used, e.g. CMOS, ECL, GaAs.
- Buses: Identify all internal system buses, their size, protocol and peak and average data transfer rate in this application. Implementation style:
- List each unique circuit and the number used in the following classes of circuits: COTS, FPGA, gate array, standard cell and full custom.
- Packaging Levels: Identify and describe the levels of packaging. For example: wirewrap back-plane, PCB pluggable module with surface mount devices, thin film MCM with ball grid array chips, ICs with various packages.
- Density: For the most dense module at the circuit board and MCM levels give the number of nets and total number of pins. For the three ICs with largest number of pins describe the package technology.
- Heat: For the highest dissipation IC give the expected maximum junction temperature under the most severe operating condition specified in the benchmark.
- Interfaces: Identify and describe system interfaces.

4.3.4 Software Products

Maintainability of software refers to the ease with which it can be understood, corrected, adapted, and enhanced. Although considered mainly for source code, it is also relevant for specification and design documents, and test plan documents. We can define three types of maintenance:

Corrective:	bug finding and fixing
Adaptive:	modifying software to properly interface with a changing environment
Enhancements:	adding new functionally to a working, successful piece of software

Maintainability is an external attribute which correlates well with certain internal, and therefore more easily measurable, attributes. These internal attributes attempt to measure characteristics of the software environment and the source code itself in the three major areas that were defined for software complexity.

In the area of comprehension, we have the classical measures of complexity such as the well known McCabe's Cyclomatic Complexity measure. This, and its variations, are useful metrics but any single metric can be distorted. We have found examples where three independent software tools for measuring Cyclomatic Complexity yielded two different values for the same module. The definition adapted for RASSP is in accordance with the IEEE Standard,

P	= number of control paths into the program
E	= number of edges (transfers of control)
N	= number of nodes
Cyclomatic Complexity	= E - N + 2P

Or, alternately, count the number of nodes in the flowgraph with two or more paths leading out from the node. The Cyclomatic Complexity value is that count plus one.[3] The Cyclomatic Complexity metric is specified in Section 4.3.4.4.

4.3.4.1 Lines of code. The number of modules in a project and the size of a module contribute to complexity. Size, as it impacts comprehension, is influenced more by the number of executable statements within the module than by the total amount of paper consumed. For this reason, when counting lines of code, count only executable lines of code as compared to the frequently seen "non comment source statements". This does not mean that comments, white space, and non-executable statements are not counted, for they are. Style is an equally important attribute.

There are two variations of the "lines of code" metric which shall be measured by the Developers for each module and for all languages used in the benchmark. The first is the usual non-comment source statements which will be measured in a manner to be consistent with the COCOMO models (LOC_COCO).The second measure will be restricted to executable lines only (LOC_EXEC). This does not include parameter definitions, type definition statements, or braces on a single line. As spreads of up to 30% within the definition have been experienced, the specific implementation for counting "lines of code" must not change, for a given language, during the course of the RASSP program. The average number of executable lines of code per module (LOC_AVG), the median (LOC_MED), and the maximum (LOC_MAX) shall also be computed as a metric. Any module having a number greater than that permitted by the style manual and not in a category permitted by the style manual (e.g., case statements) shall be considered defective.

4.3.4.2 Style. For each of the languages used in the benchmark (e. g., Ada, VHDL, FPGA) a style manual existence metric (STYLE_EXIST) with a value of zero if the manual does not exist, and 1 otherwise shall be reported. Software must have a set of uniform standards for style; if it does not, complexity increases and comprehension suffers. The style manual defines a uniform and consistent set of practices so that each module in a program appears to have been written by the same person. The following levels are assigned to the style metric (STYLE).

Level	**Characteristic**
0	Incomprehensible: each module could have been written by a different person; lack of white space and comments; lack of include files; use of embedded constants, etc. Lack of ANSI standards. Random variable naming.
1	Complex: remnants of all or most of the above
2	Moderate: use of include files, no embedded constants, nonuniform overall style
3	Churchillean: Uniform and consistent generally accepted practices according to a uniform, documented style guide consistent with the language.

Style also encompasses good programming practice. For example, the style guide may expect that an error return from a module is always checked, even though the resulting "if" test will increase the resulting complexity value. In this case, the importance of the correct style overrides the increase in complexity. This also serves to demonstrate an example of the misuse of automatic tools to make decisions without regard to human factors. Two modules with the same computed complexity could have vastly different logic structure. Consider a module such as shown in Figure 13 containing a clear sequence of submodules, each with an error return being checked according to good programming practice. Compare this with the contorted logic of the module in Figure 14 which has the same complexity value.

There will be two metrics associated with style. The first will be based on the existence of a style manual and its content. For the second, the Developer shall conduct an evaluation of all application software produced on the benchmark according to the levels (0=bad) to (3=good) in the preceding table.

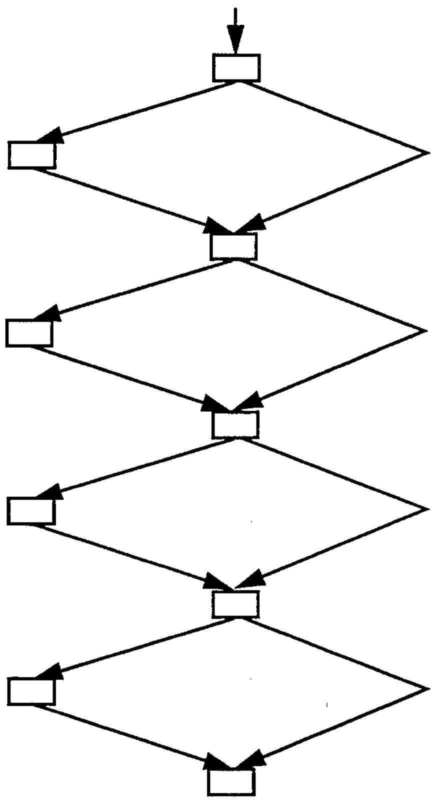


Figure 13. Simple module flowchart

4.3.4.3 Change Control. A revision control system is also an important element to reducing complexity and improving comprehension. These track a software system through its mature lifetime so that old versions can be retrieved and the changes to new versions documented. The metric for the

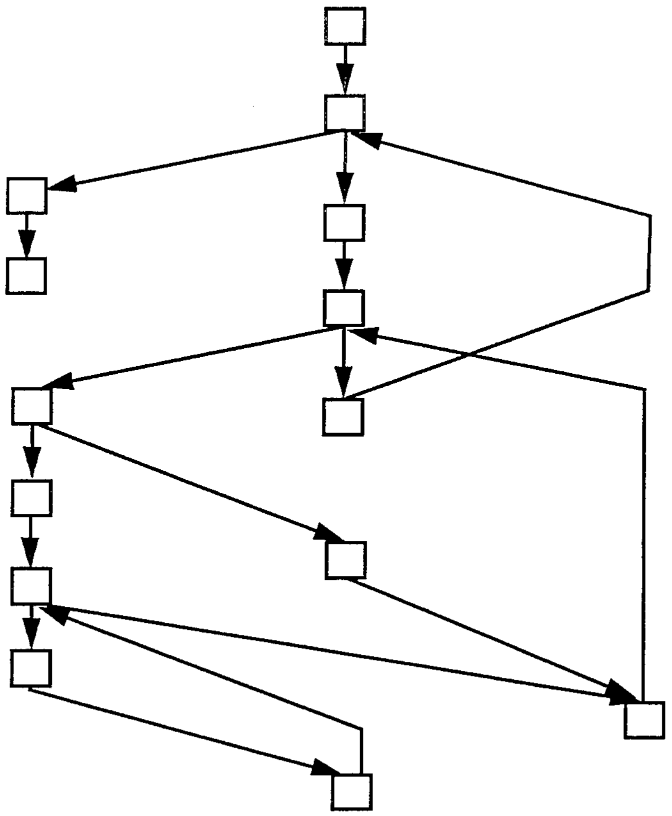


Figure 14. Complex module flowchart

software revision control system will be:

Level	**Characteristic**
0	Chaotic: source may not match binary; control by verbal agreement amongst the team
1	Primitive: Source Code Control System available free under UNIX
2	Moderate: Revision Control System or other second generation system
3	Modern: Integrated with Edit, Compile, Debug environment

The Developers shall evaluate all application software produced under the benchmark according to the scale of (0=bad) to (3=good). This metric (CNFG_MGT) shall be applied to all software.

4.3.4.4 Code Metrics. The Developers shall compute the Halstead (see Table 15) and McCabe

Table 15. Halstead Metrics

Symbol	Halstead Description
n1	number of distinct operators in a program (HAL_N_OPTOR))
n2	number of distinct operands in a program (HAL_N_OPAND)
N1	number of occurrences of operators in a program (HAL_N_OCC_R)
N2	number of occurrences of operands in a program (HAL_N_OCC_D)
n	program vocabulary (HAL_VOCAB)
N	observed program length (HAL_OB_LEN)
L	estimated program length (HAL_ES_LEN)
V	program volume (HAL_VOL)
D	program difficulty (HAL_DIFF)

(MCCABE_CCN) metrics, as defined in IEEE Standard 1061-1992 (see Section 4.3.4) on all benchmark source code. In addition, the McCabe metric shall be computed on all flow diagrams generated as part of the software design process

The control flow diagram of a module forms the basis of many complexity measures including the McCabe metric previously mentioned. This was one of the first complexity metrics and has considerable importance. There have been some validation studies and the results could fairly be described as mixed. Nevertheless, this metric has been used to good effect. Another metric in this category is that of tree impurity defined by Fenton [2]. His metric (FENTON) can be reduced to the equation

$$\frac{2\,(e-n+1)}{(n-1)\,(n-2)}, \tag{6}$$

where e is the number of edges of the graph and n is the number of nodes. Other metrics have been found that, while interesting, tend to measure the deleterious impact of the now prohibited "goto" statement. We will evaluate these options especially as available integrated into commercial products for the RASSP program. A useful source of program metrics based on the theory of flow diagrams and Fenton's work is available from the Centre for Systems and Software Engineering at South Bank University, London, but does not currently support the Ada language.

4.3.4.5 Code Defects. Two software baselines will be generated during the course of the benchmark for both the COTS and the custom design, one of which will be associated with the final prototype. The other will be generated 4-6 weeks prior. Module differences between successive baselines shall be evaluated by the Developer for differences not attributable to an intended expanded capability or to a change in requirements. Such changes shall be considered as defects.

Software testing often attempts to cover all paths through the code by branch loop testing. It is not uncommon that software works "by accident." Each code defect shall be evaluated by the Developer to ascertain the type of testing required to have detected the defect prior to release of the software (DEFECT_TST). At the end of the benchmark the Developer shall estimate the number of defects (DEFECT_RES) remaining in the software, but yet undetected, based on the observed defect density and other prior experience.

4.3.4.6 Cohesion. Cohesion is an attribute of individual modules, describing the extent to which their elements are needed to perform the same task. There are a number of classes of cohesion which form the following scale of measurement:

Level	**Characteristic**
0	Coincidental: module performs more than one function, and these are totally unrelated
1	Temporal: module performs more than one function, and these are only related by the fact that they must occur within the same time span
2	Communicational: more than one function, but all on the same body of data
3	Sequential: module performs more than one function, but these occur in an order which is described in the specification
4	Functional: module performs a single well defined function

The Developers shall evaluate all modules in the benchmark application software according to the scale of (0=bad) to (4=good) and generate the probability distribution (SCOHE_PD) associated with this data.The metrics shall be the mean, median, and peak of the corresponding probability distribution (SCOHE_AVG, SCOHE_MED, and SCOHE_MAX).

4.3.4.7 Interfaces. The interfaces have always been an area of complexity. While modern software tools prevent mis-typing across interfaces, they cannot alter the number of interfaces or the coupling in the interfaces. Coupling is a measure of the degree of interdependence between modules. There are some well established empirical relations about coupling which give the following scale of measurement:

Level	Characteristic
0	Content: a module branches into, changes data, or alters a statement in another module
1	Common: modules sharing the same global data
2	Control: a module passes a parameter to another with the intent of controlling its behavior
3	Data: modules communicate by scalar or vector data items which do not incorporate any type of control; this type of coupling is necessary for any communication between modules
4	None: totally independent

The Developers shall evaluate all pairwise modules in the benchmark application software according to the scale of (0=bad) to (4=good) and generate the probability distribution (SINTERF_PD) associated with this data.The metrics shall be the mean, median, and peak of the corresponding probability distribution (SINTERF_AVG, SINTERF_MED, and SINTERF_MAX).

4.3.4.8 Microcode. The early generations of array processors were programmed exclusively in microcode. Just as progress in hardware has improved performance over the years, so also has the software for this type of processor. By the second generation, array processors could be programmed in a high level language resembling Fortran with special features to implement such things as synchronization of a double buffered cache with main memory. Attempts, at various levels of success, were made to hide the esoterics of the microcode from the programmer through the extensive use of libraries. In the current generation, this has been carried one step further with the use of standard C or Ada code, still using extensive library modules, and less proprietary styles in making use of fundamental hardware architecture features. Today, when a specific application segment of the processing is not in the library, it can be left in the high level language and used as a compiled function if processing time constraints permit. Microcode modules which are to be added to the reuse library need to be thoroughly tested for they cannot be reexamined in the future as easily as modules written in a high level language. This is true from both a data processing point of view and a control point of view. In the latter, for example, if a module is sloppy in reading beyond the input vector address boundaries, then the consequences may be catastrophic if the operating system were to be enhanced or if new versions of the processor chip were implemented using memory management functions. Each microcode module developed for the benchmark shall be evaluated according to the metric defined in Table 16.

Table 16. Microcode Metric (MICRO)

Value	Characteristic
0	Module should have been written in a HLL. Extensive data dependencies
1	TBD
2	All paths checked, some data dependencies
3	TBD
4	No data dependencies, all paths through the module verified, communicational interfaces only

Microcode modules which are generated as part of the benchmark application software will be subject to the same metrics as any other software. The executable lines of code metric shall treat instructions executing on the same clock cycle in the same processor as a single line of code and, in addition, will generate the histogram (MICRO_HIST) of the number of instructions which are executing on each clock cycle.

4.3.4.9 VHDL. The same concept of a uniform style that is so important to common programming languages is also important to VHDL. All of the software metrics previously described in this section that are normally applied in the course of evaluation shall also be applied to the VHDL language as shall the metric associated with the use of a configuration management process. Although this language is less mature, the techniques developed by, and tested on other languages, can be brought to bear. As we have observed that the simulation time using VHDL models may be large, the wall clock time shall be measured whenever it exceeds one minute. This shall be presented in a histogram fashion (VHDL_HIST). The execution time relative to real time shall be measured for representative VHDL simulations (VHDL_SIM_TIME).

4.3.5 Documentation

A common measure of comprehension is the Flesch readability metric [5]. This is normally evaluated according to the table below which is appropriate for maintenance manuals, training manuals, or user guides This metric (FLESCH) is incorporated into the current version of WordPerfect.

Score	**Reading Difficulty**	**Grade Level**
90-100	Very Easy	4th grade
80-90	Easy	5th grade
70-80	Fairly Easy	6th grade

60-70	Standard	7th-8th grade
50-60	Fairly Difficult	Some High School
30-50	Difficult	High School to College
0-30	Very Difficult	College and up

This scale needs to be adjusted according to the intended audience. A level that is quite appropriate for one class may be barely comprehensible to a different class as shown below.

Score	Maintenance Manual	Technical Manual
90-100	Appropriate	Insulting
80-90	Appropriate	Risk losing reader's interest
30-50	Risk of losing reader's interest	Appropriate
0-30	Incomprehensible	Appropriate

As an alternative, the readability metric may use the Gunning Fog Index described below. This measures the length of sentences and the percentage of "hard" words, where a word falls into this category if it has more than two syllables. The metric (FOG) is:

$$FI = 0.4 \times \left[\frac{WRD}{SEN} + \left(\frac{HRD}{WRD}\right) \times 100\right] \qquad (7)$$

This is interpreted as shown below.

Table 17. Fog Index Interpretation

Index	**Interpretation**
FI<5	Fairly Easy
5<FI<8	Standard
8<FI<11	Fairly Difficult
11<FI<17	Difficult
17<FI	Very Difficult

5. DELIVERABLES

This section provides additional information on the deliverables required for Benchmark-1.

5.1 Processor

5.1.1 Virtual Prototype Designs

Each RASSP developer shall investigate at least two prototype processor designs, one minimizing cost to produce in prototype quantities, and the other minimizing processor power and weight. Both designs will be developed to the point where realistic estimates of performance can be made. Use of VHDL performance modeling to substantiate performance estimates is desired. Both designs must also be producible in unit quantities within the time and effort constraints established for Benchmark-2.

One of the architectural concepts investigated at the performance model level shall be selected by the Developer for evolution to a virtual prototype as described in Section 1.1.1. Insofar as possible, subject to the six month duration and 5000 hour equivalent level of effort established for Benchmark-1, the virtual prototype shall emulate the critical behavior and timing of the selected design. Although there are no hardware deliverables for Benchmark-1, all designs must adhere to the technical requirements discussed in Section 2, and the characteristics chosen for detailed modeling in the virtual prototype should be selected on the basis of these requirements.

5.1.2 Accuracy Requirements

Each RASSP developer must demonstrate a prototype processor design that meets the accuracy requirements described in Section 2.1.1. During development, however, other processor designs may achieve sidelobe and resolution performance identical to the processor described in Section 2, but may not meet the accuracy requirements of Section 2.1.1. In these cases, the RASSP Developer is encouraged to propose an alternative accuracy requirement.

The RASSP Developer must fully describe each alternative accuracy requirement and demonstrate that the alternative requirement adequately characterizes the performance of the processor. In addition, VHDL code shall be provided by the Developer so that the alternative requirement can be included in the processor testbench (Section 3). If accepted, each alternative requirement will be made available to both RASSP Developers.

5.1.3 Product Acceptance

Acceptance testing of RASSP processor prototype designs shall be performed at the Developer's site using the VHDL test bench furnished by the Benchmarker.

All virtual prototypes and associated software developed in the course of Benchmark-1, exclusive of compilers and simulators, shall be delivered to the Government after successful acceptance testing. Licenses for any commercial software libraries, operating systems, etc., exclusive of compilers and simulators, needed to execute the virtual prototype simulation shall also be delivered to the Government of their designee..

The Government and the Benchmarker shall designate witnesses for the acceptance testing, and the Government shall decide whether to accept delivery of benchmark prototype test articles based on the outcome of the acceptance testing. The Government may elect to transfer the benchmark test articles to the Benchmarker for the purpose of measuring test article compliance with all requirements included in the BTD, and assessing test article design margins.

5.2 Metrics

In addition to prototype designs and associated software, the Developer shall deliver the metrics listed in Section 5.2.1 and described in Section 4. Other metric deliverables are discussed in Section 5.2.2 through Section 5.2.5.

The metrics enumerated in Section 5.2.1 through Section 5.2.5 must be applied in a framework which considers the mode of project development as well as phase of the project (see Section 4). Each developer must, therefore, supply development mode and project phase data with each delivered metric.

All metric deliverables are due at milestone times discussed in Section 5.3.

5.2.1 Metric Deliverable Lists (MDLs).

See Tables 18 through 30 on MDLs.

5.2.1.1 PRICE-S MDL.

Table 18. PRICE-S metric deliverable list.

DESCRIPTION	SECTION	METRICS
Development CSCI	Section 4.2.2.1.1	PLTFM CMPLX INTEGI INTEGE UTIL SCON SDR SSR SRR PDR CDR TRR FCA PCA FQR OTE
Purchased CSCI	Section 4.2.2.1.2	LANG SLOC FRAC APPL INTEGE PCOST UNITS RATE RATE TIME UNIT PLTFM
Furnished CSCI	Section 4.2.2.1.3	LANG SLOC COST FRAC APPL INTEGE PLTFM

Table 18. PRICE-S metric deliverable list.

DESCRIPTION	SECTION	METRICS
Calibration CSCI	Section 4.2.2.1.4	PLTFM
		CMPLXM
		INTEGI
		INTEGE
		UTIL
		SCON
		SDR or SSR
		SRR
		PDR
		CDR
		TRR
		FCA
		FQR
		OTE
Development CSC	Section 4.2.2.1.5	INTEGI
		INTEGE
		UTIL
		SSR
		PDR
		CDR
		TRR
		FCA
Purchased CSC	Section 4.2.2.1.6	LANG
		SLOC
		FRAC
		APPL
		INTEGE
		PCOST
		UNITS
		RATE
		RATE TIME UNIT
Furnished CSC	Section 4.2.2.1.7	LANG
		SLOC
		FRAC
		APPL
		INTEGE

Table 18. PRICE-S metric deliverable list.

DESCRIPTION	SECTION	METRICS
Language	Section 4.2.2.1.8	LANG SLOC FRAC CPLX1 CPLX2 PROFAC APPL NEWD NEWC
Commercial sizing	Section 4.2.2.2.1	INTEGRATION DESIGN REVIEW CODE W/T T/D APPROACH MODULE TESTING OUTP OUTS OUTD INPF OUTF SCRF COPT INPFV COMVA LANG TARSIZ SICAL REQG FBULK

Table 18. PRICE-S metric deliverable list.

DESCRIPTION	SECTION	METRICS
Military sizing	Section 4.2.2.2.2	MIL/COM INTEGRATION DESIGN REVIEW CODE W/T T/D APPROACH MODULE TESTING OUTP ALPD INPST OUTST CSTATE INPMF INPDK INPAN COMTA FBULK REQG SICAL TARSIZ LANG
Life cycle	Section 4.2.2.3	PLTFM UTIL SSR SCHFAC DEVCST DEVU RATE TIME UNIT RATE

5.2.1.2 PRICE-M MDL.

Table 19. PRICE-M metric deliverable list.

DESCRIPTION	SECTION	METRICS
Module general A	Section 4.2.3.1.1	QTY PROTOS LENGTH, WIDTH LAYERS PLTFM NAME
Module general B	Section 4.2.3.1.2	MBINDX BTYPE BSIDE BCOST PTYPE PPINS PCOST ATCOST
PRICE-H interface	Section 4.2.3.1.3	QTYNHA INTEGE NSINT WEIGHT VOLUME BWT PWT
Module development	Section 4.2.3.1.4	ECMPLX NEWDES DESRPT
Mod. development schedule	Section 4.2.3.1.5	DSTART DFPRO DLPRO DBINDX
Mod. production schedule	Section 4.2.3.1.6	PSTART PFAD PEND MAUTO MMAT

Table 19. PRICE-M metric deliverable list.

DESCRIPTION	SECTION	METRICS
Mod. supplemental info.	Section 4.2.3.1.7	YRECON
		YRBASE
		YRTECH
		AUCOST
		ETCOST
		PRCOST
Mod. component data	Section 4.2.3.1.8	CNUM
		CNAME
		CELM
		CTYPE
		CPKG
		CPINS
		CWT
		CCOST
Microcircuit general	Section 4.2.3.2.1	QTY
		PROTOS
		LENGTH,WIDTH
		PINS
		GATES
		XSTRS
		CNAME
Micro. development	Section 4.2.3.2.2	DPLTFM
		SPLTFM
		DINDEX
		CMPLX
		NEWCEL
		DESRPT
		CADFAC
		TERAT
Micro. production A	Section 4.2.3.2.3	PROFAC
		MINDEX
		PKGFAC
		SUBFAC
		LOTQTY
		WSIZE
		FSIZE

Table 19. PRICE-M metric deliverable list.

DESCRIPTION	SECTION	METRICS
Micro. production B	Section 4.2.3.2.4	CPYLD ASMYLD OVLYLD MSKLVL DEFDEN MAUTO MMAT
Micro. dev. schedule	Section 4.2.3.2.5	DSTRT PTSRT PTEND TSTEND DEND
Micro. prod. schedule	Section 4.2.3.2.6	PSTRT PPEND PEND
Micro. supp. info.	Section 4.2.3.2.7	YRECON AUCOST
Database	Section 4.2.3.3	PLTFM YRBASE

5.2.1.3 PRICE-H MDL.

Table 20. PRICE-H metric deliverable list.

DESCRIPTION	SECTION	METRICS
Project magnitude	Section 4.2.4.1	QTY PROTOS PROSUP WT WS WECF WSCF VOL USEVOL
Customer requirements	Section 4.2.4.2	PLTFM MREL EREL
Design complexity	Section 4.2.4.3	HYBRID IC LSI VLSI MCONST MEXP MCPLXS MCPLXE AUCOST PTCOST PRCOST DTCOST
Engineering complexity	Section 4.2.4.4	ECMPLX SE
New/used design	Section 4.2.4.5	NEWST DESRPS NEWEL DESRPE

Table 20. PRICE-H metric deliverable list.

DESCRIPTION	SECTION	METRICS
Schedule impact	Section 4.2.4.6	PSF DSTART DEND DFPRO DLPRO PSTART PFAD PEND TCALD NSHIFT NFACS
Technology growth	Section 4.2.4.7	YRTECH ZTECH TECDEL
H/W and S/W integration	Section 4.2.4.8	HSINT LANG SLOC FRAC APPL CPLXM
System integration	Section 4.2.4.9	QTYNHA INTEGE INTEGS EPLANS SPLANS

Table 20. *PRICE-H metric deliverable list.*

DESCRIPTION	SECTION	METRICS
Specialized costs	Section 4.2.4.10	COST COSTTYPE CDFRAC DDRCST DDRAFT DDECST DDSIGN DSYCST DPJCST DPROJ DDACST DDATA DPRCST DTTCST DTLGTS GDTLGT PDRCST PDRAFT PDECST PDSIGN PPJCST PPROJ PDACST PDATA PPRCST PTTCST PTLGTS GPTLGT DCOST PCOST TCOST PIF UNITLC RATE RATOOL GAP GAPFAC LOTFAC OPC

Table 20. PRICE-H metric deliverable list.

DESCRIPTION	SECTION	METRICS
Other costs	Section 4.2.4.11	PTLGTS ETLG1 ETLG2 STLG1 STLG2 YRBASE YRECON DLEVE DLEVS DMULT PMULT SYSTEM ECME ECNS

5.2.1.4 PRICE-HL MDL.

Table 21. PRICE-HL metric deliverable list.

DESCRIPTION	SECTION	METRICS
Life cycle	Section 4.2.5	MTBF
		TF
		TMO
		EE
		FN
		CEND
		CPE
		CUR
		CMR
		TRE
		P
		PP
		FNSP
		CPPE
		CFIM
		CFIP
		FTSQF
		FTSQP
		TC
		CCOU
		FTSQC
		DSTART
		DEND
		PSTART
		PEND
		CUP
		CMP
		CPP
		YRECON
		YATT

5.2.1.5 SEER-SEM MDL.

Table 22. SEER-SEM metric deliverable list.

DESCRIPTION	SECTION	METRICS
Size of new code	Section 4.2.6.1.1	NEWLOC
Size of old code (non-reuse)	Section 4.2.6.1.2	OLDLOC DELLOC CHGLOC PCREDESIGN PCREIMPL PCRETEST
Size of old code (reuse)	Section 4.2.6.1.3	OLDLOC DELLOC PCREDESIGN PCREIMPL PCRETEST
Complexity	Section 4.2.6.2	COMPLEX
Personnel capabilities	Section 4.2.6.3	ANALCAP ANALEXP PROGCAP PROGLANG DEVELEXP TARGETEXP METHODEXP
Development support	Section 4.2.6.4	MODDEVEL AUTOTOOL TURNAROUNDTM TERMINALTM MULTSITE RESDEDIC RESLOC HOSTVOL METHODVOL
Product development	Section 4.2.6.5	REQVOL SPECLVL TESTLVL QALVL REHOST
Product reusability	Section 4.2.6.6	REUSELVL SWREUSE

Table 22. SEER-SEM metric deliverable list.

DESCRIPTION	SECTION	METRICS
Development environment	Section 4.2.6.7	LANGCMPLX HOSTCMPLX APPCMPLX PRACCMPLX
Target environment	Section 4.2.6.8	DISPLAY MEMORY TIME COPLEX VOLATILE SECURE
Schedule	Section 4.2.6.9	SCHEDULE
Staffing	Section 4.2.6.10	MAXPERYR MAXTOT MAXEFFRT
Probability	Section 4.2.6.11	PROB
Software requirements	Section 4.2.6.12	REQCON REQFORM REQBASE
S/W to S/W integration	Section 4.2.6.13	CONCURI ORGS EXTINTERF
S/W to H/W integration	Section 4.2.6.14	HWINTLVL UNIHW
Software maintenance	Section 4.2.6.15	YRMAIN SITES MAINGROW PERSDIFF ENVDIFF ACR MAINTOT
Add-ons	Section 4.2.6.16	EXTQA PO IVV

Table 22. SEER-SEM metric deliverable list.

DESCRIPTION	SECTION	METRICS
Avg. personnel costs	Section 4.2.6.17	SWMNG SWSR SWR SWD SWP SWQA SWCM SWDP

5.2.1.6 SEER-SSM MDL.

Table 23. SEER-SSM metric deliverable list.

DESCRIPTION	SECTION	METRICS
Pairwise	Section 4.2.7.1	PAIRWS
PERT sizing	Section 4.2.7.2	TOTLOC LIKESZ HIGHSZ
Sorting	Section 4.2.7.3	SORT
Ranking	Section 4.2.7.4	RANK

5.2.1.7 SEER-IC MDL.

Table 24. SEER-IC metric deliverable list.

DESCRIPTION	SECTION	METRICS
Product	Section 4.2.8.1	CHIPA MCMA NOMCM FSZ TPERCHIP GPERCHIP IOPERCHIP PTYPE WSZ
Mission	Section 4.2.8.2	CLASS OPENV
Program	Section 4.2.8.3	NEWD ITER CERT
Development environment	Section 4.2.8.4	DEVCAP DEVTOOL REQVOL
Product environment	Section 4.2.8.5	PROEXP PROTOOL
Program schedule	Section 4.2.8.6	STARTDEV PROQTY STARTPRO YLD
Production	Section 4.2.8.7	PRIPRO TOTPRO PCPURCH PURCHCST
Probability	Section 4.2.8.8	PROB
Economic factors	Section 4.2.8.9	DEVFEE PROFEE
Project parameters	Section 4.2.8.10	QTY STARTMO EXCHG BASEYR CSTESC DTBS

5.2.1.8 SEER-H MDL.

Table 25. SEER-H metric deliverable list.

DESCRIPTION	SECTION	METRICS
Electronic product	Section 4.2.9.1.1	TOTPCB CKTCOMP COMPCB ICPCB IOPCB CLOCK DENSE ICTECH
Electronic mission	Section 4.2.9.1.2	OPENV CLASS FLTDET FLTISO
Electronic program	Section 4.2.9.1.3	NEWD DESREP CERT HDINTLVL
Mechanical product	Section 4.2.9.2.1	WEIGHT VOLUME MATERIAL FORMCMPLX FITCMPLX CONSTR
Mechanical mission	Section 4.2.9.2.2	OPENV CLASS SRVLIFE PRESS
Mechanical program	Section 4.2.9.2.3	NEWD DESREP CERT HDINTLVL
Development environment	Section 4.2.9.3.1	DEVCAP DELTOOL REQVOL
Production environment	Section 4.2.9.3.2	PROEXP PROTOOL

Table 25. SEER-H metric deliverable list.

DESCRIPTION	SECTION	METRICS
Program schedule	Section 4.2.9.3.3	DEVSCHED DEVSTART PROTOQTY PROSTART LEARN PRIPRO PROQTY
Purchased items	Section 4.2.9.3.4	PCPURCH PURCHCST UNITCST PROB
Economic factors	Section 4.2.9.3.5	ENGRT MFGRT MATCST

5.2.1.9 SEER-HLC MDL.

Table 26. SEER-HLC metric deliverable list.

DESCRIPTION	SECTION	METRICS
Project parameters	Section 4.2.10.1	NAME OSSTART OSDUR INFLATE FYSTART COSTBY ORGHRRATE INTHRRATE DEPHRRATE
Site parameters	Section 4.2.10.2	SITEID SHIFTS SYSQTY OPSTART OPEND
Support parameters	Section 4.2.10.3	SUPSUITE CSTSUITE AVAIL
Prime mission equipment	Section 4.2.10.4	WBS NAME QTY WEIGHT OPHRS OPHRSMAT REPLACE SPARES CCR ARC MTTF CONDEMN RETESTOK
PME organization	Section 4.2.10.4.1	MTTR REPAIRRT SUPPEQ HRRT AVAIL PSECOST

Table 26. SEER-HLC metric deliverable list.

DESCRIPTION	SECTION	METRICS
PME intermediate	Section 4.2.10.4.2	MTTR TURNAROUND SUPPEQ HRRT AVAIL PSECOST NRTSRT
PME depot	Section 4.2.10.4.3	MTTR TURNAROUND SUPPEQ HRRT AVAIL PSECOST

5.2.1.10 Design process MDL.

Table 27. Design process metric deliverable list.

DESCRIPTION	SECTION	METRICS
Design process	Section 4.3.1	TOOL_USAGE TOOL_VALUE TOOL_OPEN TOOL_DACCESS TOOL_GUI TOOL_INTFCE TOOL_PROJDAT TOOL_PROJMGT TOOL_LIBMGT
Reuse	Section 4.3.1.2.1	REUSE_ENT_T REUSE_ENT_C REUSE_ORIG_T REUSE_ORIG_C REUSE_EVAL_T REUSE_EVAL_C REUSE_T REUSE_C REUSE_TRATIO REUSE_CRATIO
Software reuse	Section 4.3.1.2.1	S_REUSE_ORIG_T S_REUSE_ORIG_C S_REUSE_EVAL_T S_REUSE_EVAL_C S_REUSE_T S_REUSE_C S_REUSE_TI S_REUSE_CI S_REUSE_LOC S_REUSE_DEFECT
Defects	Section 4.3.1.3	DEFECT_FIND DEFECT_SRC DEFECT_UNDO_T DEFECT_UNDO_C DEFECT_LOST_T DEFECT_LOST_C DEFECT_TTM
Requirements traceability	Section 4.3.1.4	REQ_TRAC

5.2.1.11 Application complexity MDL.

Table 28. Application complexity metric deliverable list.

DESCRIPTION	SECTION	METRICS
Application requirements	Section 4.3.2.1.	TOTSYSOP UNISYSOP SYSOPS PROPS UNICOMOP COMPOS CONFLOW SYSRES DATARES TOTEXTINT UNIEXTINT NSTDEXTINT AVGIN PKIN ACGOUT PKOUT INSOU OUTDES LATENT DYNAMIC PRECIS
External constraints	Section 4.3.2.2	MAXSIZE MAXWGT MAXPKPOW MAXAVGPOW PRMPOW TEMP HUMID ALT CORRES SHOCK TOTCOST NRECOST

Table 28. *Application complexity metric deliverable list.*

DESCRIPTION	SECTION	METRICS
Ility requirements	Section 4.3.2.3	FLTCOV FLTLAT FLTISO MAXFLTRT MAXFLTREC SKILL DOC

5.2.1.12 Hardware product MDL.

Table 29. *Hardware product metric deliverable list.*

DESCRIPTION	SECTION	METRICS
Performance	Section 4.3.3.1	EXRATE IO DYNAMIC PKPOW AVGPOW SIZE WEIGHT COST TEST RELY AVAIL ENVIRONMENT
Complexity	Section 4.3.3.2	STORAGE GATES VLSI TECHNOLOGY IC BUSES CKTLIST PKGLIST DENSE HEAT INTERFACE

5.2.1.13 Software product MDL.

Table 30. Software product metric deliverable list.

DESCRIPTION	SECTION	METRICS
Lines of code	Section 4.3.4.1	LOC_COCO LOC_EXEC
Software style	Section 4.3.4.2	STYLE_EXIST STYLE
Software revision control	Section 4.3.4.3	CNFG_MGT
Software code metrics	Section 4.3.4.4	HAL_N_OPTOR HAL_N_OPAND HAL_N_OCC_R HAL_N_OCC_D HAL_VOCAB HAL_OB_LEN HAL_ES_LEN HAL_VOL HAL_DIFF MCCABE_CCN FENTON
Code defects	Section 4.3.4.5	DEFECT_TST DEFECT_RES
Software cohesion	Section 4.3.4.6	SCOHE_PD SCOHE_AVG SCOHE_MED SCOHE_MAX
Software interfaces	Section 4.3.4.7	SINTERF_PD SINTERF_AVG SINTERF_MED SINTERF_MAX
Microcode	Section 4.3.4.8	MICRO MICRO_HIST
VHDL simulation time	Section 4.3.4.9	VHDL_HIST VHDL_SIM_TIME

5.2.2 Tool elements

The maturity and the integration of the tools in use at each stage of the RASSP process must be measured (see Table 9). Mature tools used at the very late stages of the process, such as compilers, are known to be very stable. The same statement cannot be made about the new tools now becoming available to assist in the early stages of the design process. The amount of time spent on the telephone with each tool vendor attempting to resolve discrepancies in performance (a defect) relative to the documentation or advertised performance is a metric that must be accurately reported.

5.2.3 Reuse libraries

The use of libraries and the amount of design time that can be saved through their use is an important part of the RASSP process. This is applicable to the hardware, software, and even subsystem elements of each benchmark. The concept of reuse is applicable at all levels. To this end, the amount of time spent in exploring the reuse libraries for applicability, the time saved by not implementing an original design, and the time spent in adapting an element of the reuse library to the current benchmark must all be measured or estimated as appropriate. The amount of time spent re-evaluating the potential elements of the reuse library because the documentation is inadequate for the decision making process is a metric that must be accurately reported. Other metrics are described in Section 4.3.1.2.1

5.2.4 Documentation

Documentation should be readable and up to date. Documentation generated as part of the benchmark must be in a computer-readable format in ASCII or word processor format which can be imported into WordPerfect 6.0. A PostScript file is not considered to be computer-readable. The Flesch readability metric, or equivalent, will be applied by the Developers to all, or at least some portion, of the documentation. For documentation supplied by the tool vendors, samples of the text will be used to generate the Flesch readability metric. Online documentation must be evaluated for its pertinence, accuracy, and level of detail.

5.2.5 Software metrics

During the course of each benchmark, software baselines shall be created by the Developer as a deliverable item and are due at the milestones discussed in Section 5.3. For the purposes of this benchmark, software specifically includes VHDL code. A baseline is not intended to be comprehensive or a final version but is intended to represent a working package for some subset of the overall task. The Developer shall apply the set of software metrics described in the section on Software Products (Section 4.3.4) to each version of the baseline software. The benchmark evaluation process will apply a set of metrics to all baseline software delivered with the benchmark [4] as appropriate. During the course of the benchmark, baseline software which has been identified as lacking in specified features intended for a later version will not be considered to contain a defect for these omissions.

Whatever style guides a Developer imposes for software development at the beginning of each benchmark are considered deliverable items. It is the intent to track these over the course of the RASSP program. A description of the configuration management tools that are in place at the beginning of a benchmark is a deliverable item. During the course of the benchmark, all occurrences of bypassing the configuration management protocols must be reported by e-mail, or a suitable equivalent, to a central repository. The contents of the repository become a deliverable.

The developer shall indicate the perceived level of conformance to the SEI Capability Maturity Model at the beginning of each benchmark. Note that this is not intended to be a formal CMM review. This level is to be based exclusively on the software methodology in place for the RASSP program and is not to be based on methodology available in other parts of the Developer's company, no matter how advanced it may be.

The contents of code inspections and their results form part of the deliverables. This includes, of course, structure charts and flow diagrams.

5.3 Milestone Reports

Up to date software, prototype design, trade analyses, and metric deliverables shall be provided at each milestone reached during each benchmark cycle. The milestones are to be defined by the Developer and clearly described in the response to this Benchmark Technical Description. Four milestone reviews are required. The first review should correspond to the time at which the system requirements are defined. The second milestone review should correspond to the time at which architectural trade-offs have been completed and a preferred architecture has been selected. The third and fourth review are left to the discretion of the Developer. One basis for choosing the third and fourth milestones is to key the third to successful execution of an initial virtual prototype, and the fourth to expiration of the six month benchmark time period.

However, a preferred approach is to key the third and fourth milestones to the Developer's design process. As an example, assuming a spiral development model for the RASSP process[11], the third and fourth milestones might be keyed to completion of a loop around the spiral. Figure 15 shows a nominal spiral development model for Benchmark-1, with three spiral cycles occurring within a six-month benchmark cycle. For Benchmark-1 the development and trade-off of at least two designs might be viewed as the first loop around the spiral.

Each spiral cycle encompasses four phases of development. The processes starts with a planning phase. However, for Benchmark-1 it is assumed that initial design planning occurred with Benchmark-0 and is represented by the dotted line in Figure 15. Thus, Benchmark-1 formally begins with a requirements review. The phase-1 consists of developing preliminary designs. These designs are evaluated within the phase-2 after which a single best design is identified. Phase-3 continues design development while design deficiencies, if any, are identified. Phase-4 uses design information gather during the three preceding phases to prepare for the next spiral cycle.

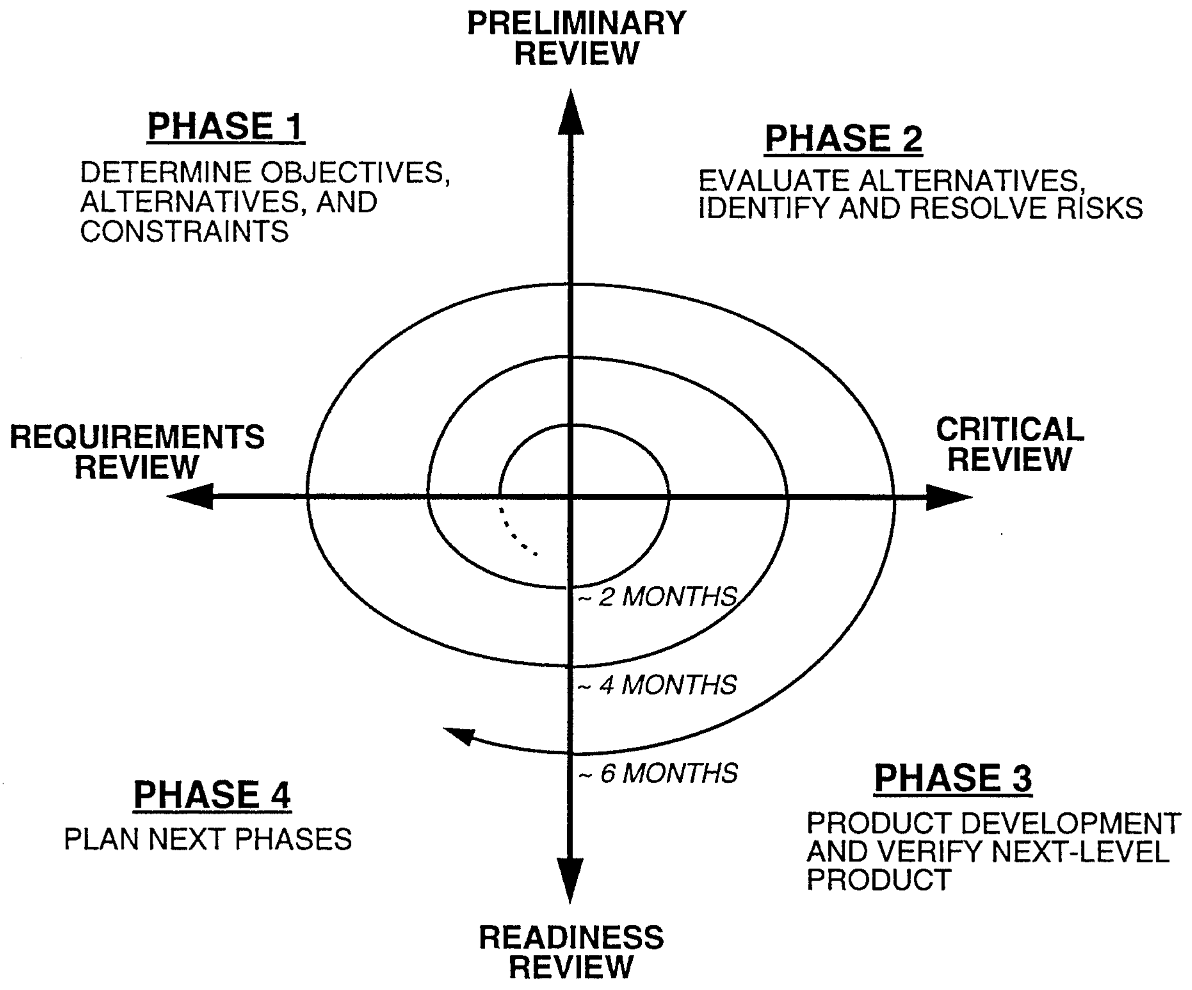

PRELIMINARY REVIEW
PHASE 1
DETERMINE OBJECTIVES, ALTERNATIVES, AND CONSTRAINTS
PHASE 2
EVALUATE ALTERNATIVES, IDENTIFY AND RESOLVE RISKS
REQUIREMENTS REVIEW
CRITICAL REVIEW
~ 2 MONTHS
~ 4 MONTHS
~ 6 MONTHS
PHASE 4
PLAN NEXT PHASES
PHASE 3
PRODUCT DEVELOPMENT AND VERIFY NEXT-LEVEL PRODUCT
READINESS REVIEW

Figure 15. Spiral development model for Benchmark-1.

Reporting, software, prototype design, and metric deliverables shall be due at the completion of each phase of each spiral cycle. Figure 16 shows a Gantt chart for the spiral model of Figure 15.

At the start of Benchmark-1, each Developer shall provide details on the specific development model to be used throughout the benchmark cycle, including descriptions of the development phases. At the start of Benchmark-1, each developer shall generate a detailed schedule of milestones and deliverables (i.e., reports, metrics) for the entire benchmark cycle. At each milestone, each Developer will provide actual schedules of activity to be compared with those generated at the start of Benchmark-1.

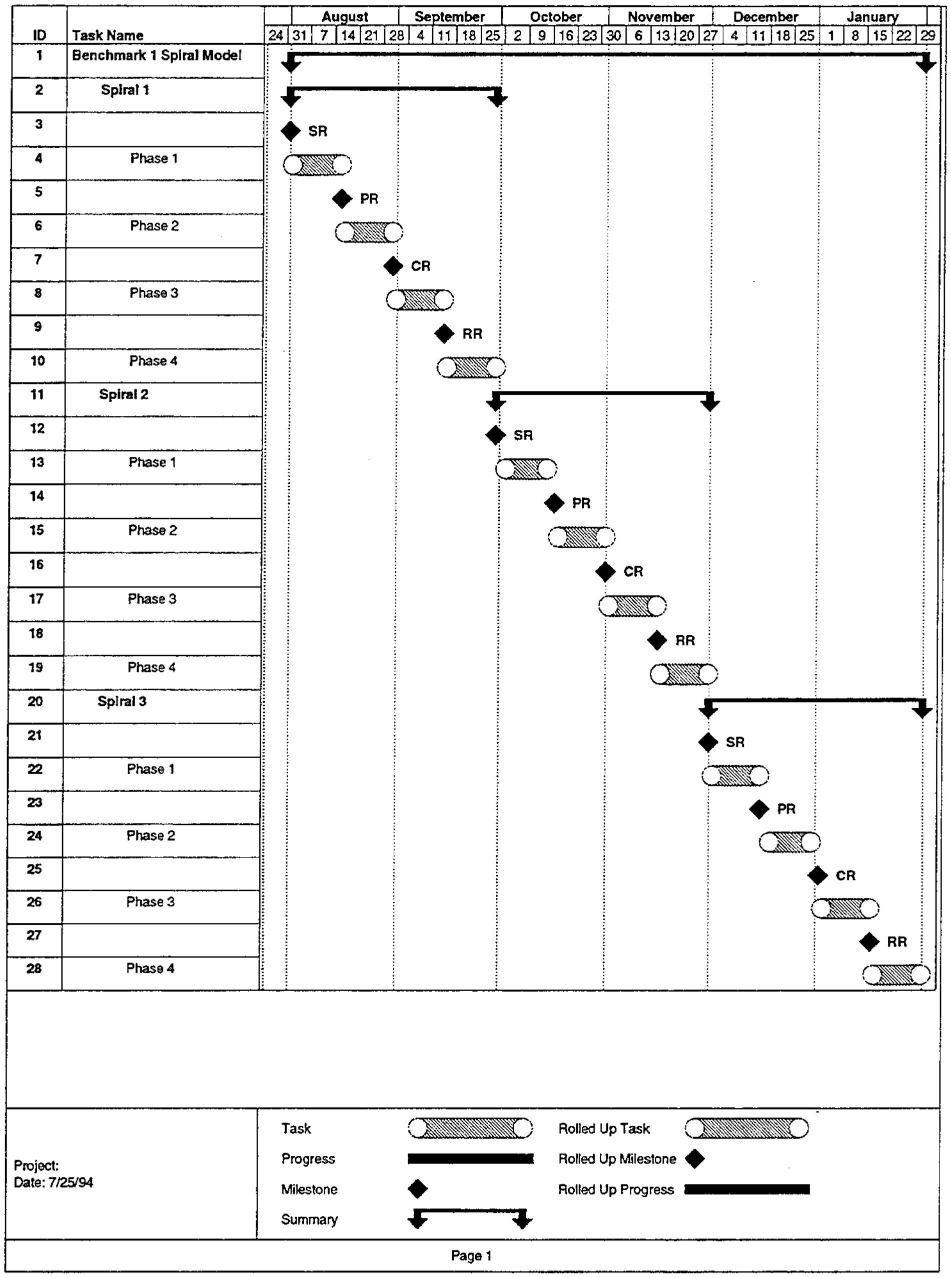


Figure 16. Gantt chart for Benchmark-1 milestone reports.

Informal reports are due at each milestone that correspond to the deliverables required for each milestone. In addition, these reports shall review progress and problems encountered since the last milestone review. Formal milestone reports shall also incorporate comprehensive management data including actual and projected costs and schedule for the benchmark. Formal milestone reports shall include a comprehensive representation of the design database at the time the milestone was reached. Formal milestone reports will not be required more frequently than once per month over the duration of Benchmark 1, except in the event that the benchmark execution is completed in substantially less time than originally estimated by the Developer. All milestone reports shall be sufficiently comprehensive so that, when taken as a whole, they provide a detailed description of the progress and problems encountered in completing execution of the benchmark.

5.4 Electronic Reporting

Unless otherwise agreed upon in writing, the Developer shall supply all non-hardware deliverables and reports in the following electronic formats. Where multiple formats are noted, the Developer can select the format most appropriate to the data item. Style and format files must also be supplied whenever either is required to view or print a data item.

- Schedules - Microsoft Project or a compatible format
- Reports/Documentation - WordPerfect, Microsoft Word, or Framemaker
- Spreadsheet - Microsoft Excel or a compatible format
- Project Data - Both native tool format and project-wide database format
- HOL/HDL Source code - ASCII machine-readable format

The digital data may be provided via an Exabyte model 8200 or 8500 uncompressed tar format 8mm tape, or via an FTP site accessible over the Internet. Wherever requested in the BTD for a given benchmark, the deliverables shall also be supplied in printed form. Password protection may be used fort security at the Developers' option.

5.5 Benchmark Personnel Reporting

During the execution of the benchmark, each of the individuals may be required to participate in informal, but regularly scheduled, progress review meetings. The project review meetings shall nominally be held on a weekly basis, but no less frequently than biweekly, with major contributors to the benchmark execution summarizing the focus of their work over the prior week. The size of the benchmark execution team and the scope of the benchmarks is anticipated to be small enough that weekly meetings will require only 1-2 hours to complete. The main purpose of these meetings is to maintain a running account of progress on the benchmark, including milestones, problems, schedule changes, etc. Ideally, the weekly or biweekly meetings will be held in connection with normal project management meetings, rather than becoming an additional set of meetings. RASSP program meetings and reviews, including Benchmark milestone reviews, may be substituted for the benchmark meetings by mutual agreement between the Developer and the Benchmarker.

All person-hours expended on the benchmark execution must be reported and associated with tasks in the WBS. The reporting must be sufficiently detailed to distinguish between in-cycle and out-cycle activities. The desired precision of task time reporting is .5 hours.

6. DEVELOPER RESPONSE

This section provides additional detail regarding the response the Developer shall provide to BTD-1.

6.1 Benchmark Execution Check List

For Benchmark 1, the Developer shall include in the response to the BTD a Benchmark Execution Check List (BECL). The BECL shall be based on the RASSP process steps which the Developer envisions applying to execute the benchmark. For each major process step, the Developer shall provide the following information:

1. Cost
2. Schedule
3. Tools utilized
4. Caliber of individual(s) required to execute the process

The BECL can also be organized according to the deliverables (products) required in BTD-1, but in this case, the process steps and cost associated with the development of each deliverable must be indicated where appropriate. For example, since performance models are a deliverable, the process steps and tools used to produce the performance models must be indicated. In the case of metric deliverables, the costs should be broken out on the basis of the metric categories defined in Table 18 through Table 30.

The BTD includes points of contact at the Benchmarker's organization for the purpose of addressing technical questions regarding the BTD, however, all questions submitted to the Benchmarker shall also be submitted simultaneously to the cognizant Government COTR or his designee.

The Developer shall respond with a comprehensive estimate of the cost to execute BTD-1. The Developer shall include a WBS and associated schedule for the tasks in the WBS, along with a list of all the individuals assigned to work on the Benchmark more than an average of one day a week. The level of detail shown in the WBS and schedule shall be sufficient to identify and briefly describe the distinct steps in the Developer's RASSP design process, and shall conform to specific formats and reporting details called for in this BTD. For each entry in the BECL, the total estimated cost of executing that part of the RASSP process shall be required. An indication shall be provided of the cost and schedule impact on the remaining process steps of deleting a particular process step. The categories of impact are:

- None
- Modest
- Significant
- Essential

6.2 Tool Status Information

At the outset of Benchmark 1, along with the schedule and cost estimate, the Developer shall provide a list of all of the electronic design automation (EDA) tools available in the RASSP system, an indication of which tools are likely to be used, and a description of the RASSP design process supported by the tools. The tool and process description shall include, as a minimum, the following information, and shall be provided in written form and in one of the electronic formats described in Section 5:

- The association between the tools and the RASSP process steps
- The integration status of each tool including:
  - Revision number of the tool
  - Interfaces to other tools
  - Level of integration as described in Section 6.3
- Minimum host machine resources required to effectively use each tool including:
  - Minimum host memory configuration for executable
  - Disk resources required
  - Representation of the minimum acceptable CPU performance (e.g. Specmarks)
- Platforms on which each tool is supported
- Purchase and maintenance costs for each tool
- The minimum skill category or area of specialization required to effectively use each tool. Example tool and skill categories are given below:

TOOL	SKILL CATEGORY
Word Processor	Secretary/Technical Writer
Architecture Trade-off	System Analyst
Ada Compiler	Programmer
Thermal Design	Mechanical Engineer
VHDL Simulator	Digital Designer
Schematic Entry	Technician

In order to visualize the degree of tool integration within the RASSP design environment, the equivalent of a 2-D matrix (N^2 chart) of the available tools will be created and the level of integration which exists between all pairwise combinations of tools will be entered as a number at the row and column intersection of the tool pair. The level of tool integration shall be supplied by the Developers and verified by the Benchmarker.

6.3 Benchmark Personnel Reporting

A list of all the individuals projected to work on Benchmark 1 an average of one or more days a week must be provided at the initiation of a benchmark. The list should indicate the title and job category of each of the individuals, along with a description of their familiarity with both the benchmark application and the RASSP tools and processes. Personnel changes made during the course of the benchmarking by the Developer shall be reported to the Benchmarker at the time the changes are made.

REFERENCES

1. Henry, J.C., "The Lincoln Laboratory 35 GHz airborne polarimetric SAR imaging system," *IEEE National Telesystems Conf.*, Atlanta, GA, 26-27 March, 1991, p.353.

2. Fenton, N.E., *Software Metrics, A Rigorous Approach*, Chapman and Hall 1991.

3. Whitty, R. and R. Lockhart, *Structural Metrics*, Goldsmiths' College, London 1992.

4. *QUALMS User's Guide*, South Bank University, London 1993.

5. Flesch, R.F., *The Art of Readable Writing*, Harper & Row, 1974.

6. Wong, R., "Cost modeling," Ch. 20 in *Space Mission Analysis and Design*, 2nd Edition, Ed. by W.J. Larson and J.R. Wertz, Kluwer Academic Publ., Norwood, MA, 1992.

7. Boehm, B.W., *Software Engineering Economics*, Pretice-Hall, Inc., Englewood Cliffs, NJ, 1981.

8. Davis, J.M., "Planning and estimating software," *CASE Outlook 89*, No. 2, pp. 23-34, 1989.

9. Martin Marietta PRICE Systems, *PRICE Reference Manuals*, Moorestown, NJ, 1981.

10. Galorath Associates, Inc., SEER Technologies Division, *SEER User Manual*s, Los Angeles, CA, 1993.

11. Boehm, B.W., "A spiral model of software development and enhancement," *IEEE Computer*, pp. 61-72, May 1988.

APPENDIX A

Convolution Kernels

Expressions used for determining the convolution kernels can be developed by considering the complex received LFM (linear frequency modulation) signal transmitted at time t_i and received at time t,

$$s(t) = Ae^{j\left[2\pi f_o(t-\tau_r) + \pi K(t-t_i-\tau_r)^2\right]}, \tag{A.1}$$

where f_0 is the transmit center frequency (in Hz), A is the signal magnitude, K is the LFM slope (in Hz/sec), and τ_r is the round-trip propagation time to the target. The de-ramp signal for s(t) is given by,

$$d(t) = e^{-j\left[2\pi f_o(t-\tau_c) + \pi K(t-t_i-\tau_c)^2\right]} \tag{A.2}$$

where τ_c is the round-trip propagation time for a target at a reference range R_0,

$$\tau_c = \frac{2R_0}{c}. \tag{A.3}$$

The resulting de-ramped received pulse is given by,

$$x(t) = s(t)d(t) = Be^{j\phi(t)}, \tag{A.4}$$

where,

$$\phi(t) = \pi K\left[\tau_r^2 - \tau_c^2 - 2(\tau_r - \tau_c)(t-t_i)\right] - 2\pi f_0(\tau_r - \tau_c). \tag{A.5}$$

In range compression, x(t) is weighted and sampled at time,

$$t = t_i + \tau_c + T_S[n-m], \tag{A.6}$$

where

$$n = 0, \ldots, N-1, \tag{A.7}$$

T_S is the sampling interval, and N is the total number of range samples. The weighted and sampled version of x(t) is given by,

$$y(n) = W(n-m)x(n) = W(n-m)Be^{j\phi(n)} \tag{A.8}$$

where,

$$\phi(n) = \pi K\left[(\tau_r-\tau_c)^2 - 2(\tau_r-\tau_c)(n-m)T_S\right] - 2\pi f_0(\tau_r-\tau_c) \qquad \text{(A.9)}$$

and W(n-m) is a real-valued weighting function symmetric about n=m. The range-compressed pulses are computed as the DFT of y(n) with zero-padding, so that a compressed pulse is given by

$$DFT\{y(n)\} = DFT\{W(n-m)e^{-j2\pi KT_S(\tau_r-\tau_c)(n-m)}\} \times$$

$$\times Be^{j\left[\pi K(\tau_r-\tau_c)^2 - 2\pi f_0(\tau_r-\tau_c)\right]} . \qquad \text{(A.10)}$$

Moreover,

$$DFT\{W(n-m)e^{-j2\pi KT_S(\tau_r-\tau_c)(n-m)}\} = DFT\{W(n)e^{-j2\pi KT_S(\tau_r-\tau_c)n}\} \times$$

$$\times e^{j2\pi(mi/N)} , \qquad \text{(A.11)}$$

where i is the sample in the transform domain. Because $W(n)\exp[-j2\pi KT_S(\tau_r-\tau_c)n]$ is a conjugate-symmetric function of n, its DFT is real-valued. The phase term, $\exp[j2\pi(mi/N)]$, is constant for all pulses. For the case of m=N/2, the phase term reduces to $(-1)^i$. As a result, the phase of the compressed pulse, Φ, is given by the right-most exponential in (A.10),

$$\Phi(n) = \pi K(\tau_r-\tau_c)^2 - 2\pi f_0(\tau_r-\tau_c) . \qquad \text{(A.12)}$$

The convolution kernels for azimuth compression, $\exp[j\Phi_{CONV}]$, are formed on the basis of (A.12). The Aux variable SLTRNG gives the slant range to the center of the first frame of the pass, R_0. Reference ranges for the 31 kernels are those between the ranges of

$$r_0 = R_0 - 15.5d_r \qquad \text{(A.13)}$$

and

$$r_0 = R_0 + 14.5d_r \qquad \text{(A.14)}$$

at a range interval of d_r, where d_r is 1/16th the length of the range window. The length of the current range window is 468.4 m, so that d_r=29.3 m. As a result, 16 kernels are needed to process all 2048 range-gates of the processing array and each kernel is used for 128 contiguous range-gates. The kernels used for a particular processing array are determined from the center range of the input frame to the array, R, where R comes from the Aux variable SLTRNG. That is, the 16 convolution kernels are used whose reference range lies between

$$r_0 = R_0 + \left(-7.5 - INT\left[\frac{R_0 - R}{d_r}\right]\right)d_r \tag{A.15}$$

and

$$r_0 = R_0 + \left(7.5 - INT\left[\frac{R_0 - R}{d_r}\right]\right)d_r \tag{A.16}$$

inclusive, where $INT[\ \ ]$ indicates the nearest integer. A frame having $R=R_0$ would use 16 convolution kernels whose reference range lies between $R_0 - 7.5d_r$ and $R_0 + 7.5d_r$, inclusive.

In evaluating (A.12) for a particular convolution kernel, we note that τ_r changes from PRI-to-PRI but τ_c does not. Consider a constant velocity SAR platform that transmits a pulse at a constant spatial interval d_x; recall that d_x=0.2287 m for the current system. If $\tau_r = \tau_c$ at the PRI where the target is broadside to the SAR platform, the range to the target at any other PRI is approximately,

$$RANGE|_{PRI=k} = r_0 + \frac{(kd_x)^2}{2r_0} \tag{A.17}$$

where k is the (integer number) difference between the PRI being processed and the PRI of target-broadside. As a result, τ_r is approximated by,

$$\tau_r \approx \left(\frac{2}{c}\right)\left[r_0 + \frac{(kd_x)^2}{2r_0}\right]. \tag{A.18}$$

The phase of the convolution kernel is the complex conjugate of Φ, so that the kernel is given by,

$$e^{j\Phi_{CONV}} = e^{-j\Phi} \tag{A.19}$$

where Φ is given by (A.12), and τ_c and τ_r are given by (A.3) and (A.18), respectively. The convolution kernel (i.e., Φ_{CONV}) must be calculated for all pulses within the length of the synthetic aperture, where

$$\text{APERTURE LENGTH} = \frac{\lambda r_0}{2RES}. \tag{A.20}$$

At a center frequency f_0 = 33.56 GHz, a resolution of RES = 0.3 m and a range of r_0 = 7.26 km yields an aperture length of 108.09 m. Because d_x = 0.2287 m, the minimum number of pulses in the convolution kernel is 473. Currently, 512 pulses are used in calculating the convolution kernels.

APPENDIX B

Taylor Weight

The Taylor weights required for creating convolution kernels and performing real-time range compression are calculated during initialization processing. Inputs for these calculations are the number of weight samples (N), the maximum sidelobe level in dB (SLL), and the number of near-in sidelobes $\bar{n}$; currently $\bar{n} = 6$ and SLL = -30 dB. The weight at the ith sample is calculated as,

$$w(i) = 1 + 2 \sum_{m=1}^{\bar{n}-1} F_m \cos(2\pi m x_i) \tag{B.1}$$

where i goes from 1 to N and

$$x_i = \begin{cases} \dfrac{i - \dfrac{N+1}{2}}{N} & \text{for N odd} \\ \dfrac{i - \dfrac{N+2}{2}}{N} & \text{for N even} \end{cases} . \tag{B.2}$$

The coefficient F_m in (B.1) is given by,

$$F_m = \frac{0.5(-1)^{m+1} \prod_{n=1}^{\bar{n}-1} \left[1 - \dfrac{s_p m^2}{A^2 + \left(n - \frac{1}{2}\right)^2} \right]}{\prod_{\substack{p=1 \\ p \neq m}}^{\bar{n}-1} \left(1 - \dfrac{m^2}{p^2}\right)} , \tag{B.3}$$

where s_p is a variant of the Taylor pulse widening factor[1] and is given by,

1. See *Radar Signals* by Cook and Bernfeld, Academic Press, 1967.

$$s_p = \frac{A^2 + \left(\bar{n} - \frac{1}{2}\right)^2}{\bar{n}^2}. \tag{B.4}$$

The factor A in (B.3) and (B.4) is related to the maximum sidelobe level by,

$$A = \frac{\cosh^{-1}\left(10^{-SLL/20}\right)}{\pi}. \tag{B.5}$$

However,

$$\cosh^{-1} = \ln\left(x + \sqrt{x^2 - 1}\right), \tag{B.6}$$

so that

$$A = \frac{\ln\left(s_f + \sqrt{s_f^2 - 1}\right)}{\pi}, \tag{B.7}$$

where

$$s_f = 10^{-SLL/20}. \tag{B.8}$$

APPENDIX C

Fiber Optic Module

TriQuint SEMICONDUCTOR

Hot Rod™
Fiber Optic Cards – Short Wavelength
HRC–200FS, HRC–500FS, and HRC–800FS

General Description

The HRC–200FS, HRC-500FS and HRC–800FS fiber-optic Hot Rod cards provide a complete bidirectional node for transmitting and receiving 40-bit parallel data words across fiber-optic media at maximum rates of 200, 500, and 800 Mbits/sec. (250, 625, and 1000 MBaud). These cards are ideal for use in high-performance systems where data communications is the bottleneck to system performance.

The interface to the Hot Rod card is designed for flexibility. The convenient 120-pin high-density connector offers a small physical interface. The transmitter and receiver sections have separate data and control signals, allowing them to operate simultaneously and independently. Only a single +5V power supply is required.

Several user options allow for a broad range of operation. Data transmission can be selected to be 200, 400, or 800 Mbits/sec. for the HRC–800FS, 250 and 500 Mbits/sec. for the HRC–500FS, and 200 Mbits/sec for the HRC–200FS. The on-board optical data links transmit and receive data across multi-mode fiber-optic media. The optics on the cards are designed using short wavelength CD laser technology.

The Hot Rod card is an excellent vehicle for all stages of development. Because it is a complete solution, it can shorten the design cycle considerably during prototyping. Its compact size and proven design make it an excellent production solution.

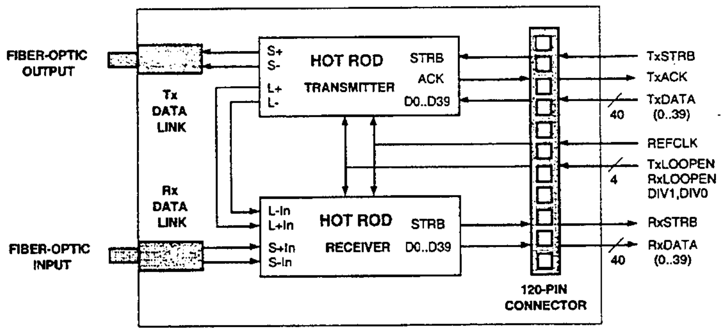
Typical Applications

- High–speed networks
- Point–to–point communications
- Bus extenders
- High–bandwidth digital video transmission

Features

- Complete fiber-optic communications node
  - — Hot Rod transmitter
  - — Hot Rod receiver
  - — On-board Optical Data Links
  - — CD laser technology
  - — Transmit and receive up to 300 meters at 800 Mbits/sec.
  - — Transmit and receive up to 1000 meters at 500 Mbits/sec.
- Selectable data rates:
  - — Speeds of 200, 400, 800 Mbits/sec. for HRC–800FS
  - — Speeds of 250, 500 Mbits/sec. for HRC–500FS
- Multi-mode (50/125 or 62.5/125) fiber compatibility
- Compatible with 32-bit microprocessor systems
  - — 40-bit TTL input (transmit) bus
  - — 40-bit TTL output (receive) bus
- Loopback diagnostic capability
- Single +5V supply
- Operable with the Hot Rod Development System (HRDS)
- Bit Error Rate (BER) ≤ 10^{-12}
- Link status monitoring capability
- 120-pin high-density connector
- ST-type fiber-optic connectors
- Compact size (approximately 3³/₄" x 5")
- Commercial temperature range

Block Diagram

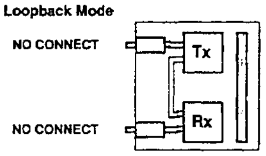


TQS Computing and Networking Division • 2300 Owen Street • Santa Clara, CA 95054 • (408) 982-0900 • FAX (408) 982-0222

Operating Modes

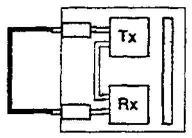
The HRC-xxxFS fiber-optic Hot Rod cards can operate in three modes: on-board loopback, fiber loopback, and standard fiber. These are outlined below.

On-Board Loopback Mode



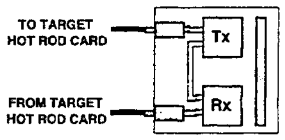
The HRC-xxxFS Hot Rod cards may be operated without any fiber connection. By asserting both LOOPEN signals HIGH, the transmitter device will write directly to the local receiver device. This diagnostic capability is a useful power-up and first test exercise.

Fiber Loopback Mode



One Hot Rod card is capable of testing various fibers. As in Mode 1 above, the transmitter sends to the local receiver. In this case, however, the signal is carried through a fiber that has been connected from the transmit link to the receive link.

Standard Fiber Mode



The HRC-xxxFS can also be used as a bidirectional fiber-optic communications node. The transmit link is connected (via fiber) to a target Hot Rod card. Similarly, data is received from a target Hot Rod card.

Data Rates

The user data rate is selected using the frequency of the reference clock (REFCLK) and the value of the DIV1 and DIV0 input control signals. The Hot Rod devices are specified to run with a REFCLK of 20 MHz and 25 MHz. The word rate can then be divided down (by a factor of 2 or 4), using the DIV signals as indicated in the table below.

The REFCLK is an external TTL-compatible signal which is input through the connector. This signal must be within the limits detailed in the AC Specifications of the Hot Rod device data sheet. The various allowable frequencies and configurations are:

DIV 1	DIV 0	WORD RATE (Mwords/sec.)	BIT RATE (Mbits/sec)	REFCLK (MHz)
1	1	20	800	20
1	0	10	400	20
0	1	5	200	20
1	0	12.5	500	25
0	1	6.25	250	25

Note: These are the only valid configurations for the HRC-xxxFS Hot Rod card.

Optical Specifications (Temp = 0-60°C)

	TRANSMITTER		RECEIVER		
	MIN	MAX	MIN	MAX	UNITS
Center Wavelength	830	870	—	—	nm
Spectral Width	—	15	—	—	nm
Average Power	–3.5	0	*	0	dBm
Rise/Fall Time	—	0.4 ÷ B[1]	—	0.5 ÷ B[1]	ns
Extinction Ratio	5 : 1	—	—	—	

Note: [1] B = Baud Rate = 1.25 x Bit Rate

* See Max Link Loss table below

Cable Plant Specifications

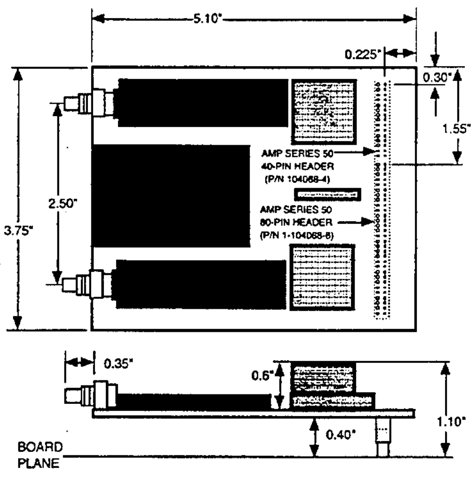
	FIBER TYPE				
	50/125		62/125		
	MIN	MAX	MIN	MAX	UNITS
Core Diameter	—	50*	—	62.5*	microns
Cladding Diameter		125*		125*	microns
Data Rate (–800)	200	800	200	800	Mbps
Data Rate (–500)	250	500	250	500	Mbps
Modal Bandwidth	>400	—	>200	—	MHz•km

* Nominal

	DATA	MAX. DISTANCE		MAX LINK LOSS*	
	RATE	50/125	62/125	50/125	62/125
HRC–800FS	200	2000 M.	1000 M.	6	7
	400	1000 M.	1000 M.	5	6
	800	300 M.	300 M.	3	3
HRC–500FS	250	2000 M.	1000 M.	6	7
	500	1000 M.	300 M.	4	6
HRC–200FS	200	2000 M.	1000 M.	6	7

*Tested with 2 sets of connectors.

Mechanical Dimensions


5.10"
0.225"
0.30"
1.55"
AMP SERIES 50 40-PIN HEADER (P/N 104068-4)
AMP SERIES 50 80-PIN HEADER (P/N 1-104068-8)
2.50"
3.75"
0.35"
0.6"
1.10"
0.40"
BOARD PLANE

Hot Rod Development System

In order to facilitate evaluation of the Hot Rod chipset, Hot Rod cards, and the capabilities of various fibers, TriQuint offers the Hot Rod Development System. The Development System HRDS-800FS, for example, comes with the HRC-800FS Hot Rod interface card, and provides a variety of capabilities, from test pattern generation to Bit Error Rate counting and automatic logging of test results. The Development System includes EPROM-based, menu-driven software for running many different tests, and it may also be user-programmed for special applications.

Power Supply Specifications

The board should be supplied by a high-quality, computer-grade power supply capable of at least 1.5 A at +5 V (4.75V min., 5.25 V max.).

Edge Connector Specification

TxD2	61	1	TxD0
TxD3	62	2	TxD1
TxD6	63	3	TxD4
TxD7	64	4	TxD5
TxD10	65	5	TxD8
TxD11	66	6	TxD9
TxD14	67	7	TxD12
TxD15	68	8	TxD13
TxD18	69	9	TxD16
TxD19	70	10	TxD17
TxD22	71	11	TxD20
TxD23	72	12	TxD21
GND	73	13	GND
GND	74	14	GND
TxD26	75	15	TxD24
TxD27	76	16	TxD25
TxD30	77	17	TxD28
TxD31	78	18	TxD29
TxPWR	79	19	TxPWR
TxPWR	80	20	RESERVED
TxD34	81	21	TxD32
TxD35	82	22	TxD33
TxD38	83	23	TxD36
TxD39	84	24	TxD37
TxDIV1	85	25	TxACK
TxDIV0	86	26	TxLOOPEN
TxSLAVE	87	27	RxLOOPEN
TxSTRB	88	28	REFCLK
GND	89	29	Tx1XCLK
GND	90	30	Tx2XCLK
Rx2XCLK	91	31	GND
Rx1XCLK	92	32	GND
RESERVED	93	33	RxSTRB
$\overline{\text{SIGDET}}$	94	34	RESERVED
RxDIV1	95	35	RxSYNC
RxDIV0	96	36	RxERROR
RxD2	97	37	RxD0
RxD3	98	38	RxD1
RxD6	99	39	RxD4
RxD7	100	40	RxD5
RxPWR	101	41	RxPWR
RxPWR	102	42	RESERVED
RxD10	103	43	RxD8
RxD11	104	44	RxD9
RxD14	105	45	RxD12
RxD15	106	46	RxD13
GND	107	47	GND
GND	108	48	GND
RxD18	109	49	RxD16
RxD19	110	50	RxD17
RxD22	111	51	RxD20
RxD23	112	52	RxD21
RxD26	113	53	RxD24
RxD27	114	54	RxD25
RxD30	115	55	RxD28
RxD31	116	56	RxD29
RxD34	117	57	RxD32
RxD35	118	58	RxD33
RxD38	119	59	RxD36
RxD39	120	60	RxD37

Ordering Information

	DATA RATE (Mbits/sec)				
	200	250	400	500	800
HRC–800FS	✓		✓		✓
HRC–500FS		✓		✓	
HRC–200FS	✓				

Handling Precautions

1. High electrostatic fields can permanently damage the device. Normal handling precautions for electrostatic-sensitive devices should be taken (as CMOS).
2. Semiconductor lasers are easily damaged by overloading or by current surges. Appropriate transient protection precautions should be taken.

Safety Precautions

DANGER

Invisible LASER Radiation
Avoid direct exposure to beam

Peak Power 1 mW
Wavelength 880 nm

Class IIIb LASER Product

EMISSION DIRECTION

WARNING!
LASER RADIATION – This device in operation produces invisible electromagnetic radiation which may be harmful to the human eye.

TriQuint will not be liable for any damages arising from the use of this Laser Diode-based product.

Typical Test Results

Board	Medium Multi-Mode Fiber	Type	Distance	No. of Bits Transferred	Errors	Upper BER Estimate (95% Confidence Level)
500 Mbaud (–400)	50/125μm	CD Laser	2.250 Km	2.95E+13	None	1.04125E–13
	62.5/125μm	CD Laser	1.151 Km	8.85E+12	None	3.4697E–13
625 Mbaud (–500)	50/125μm	CD Laser	1.143 Km	9.45E+12	None	3.24893E–13
	62.5/125μm	CD Laser	1000 feet	4.6E+12	None	6.67538E–13
1000 Mbaud (–800)	50/125μm	CD Laser	1000 feet	5.18E+13	None	5.92887E–14
	62.5/125μm	CD Laser	1000 feet	6.5E+13	None	4.72261E–14

TriQuint Semiconductor, Inc.
Computing and Networking Division
2300 Owen Street
Santa Clara, CA 95054
(408) 982-0900

TriQuint Semiconductor, Inc.
3601 SW Murray Boulevard
Beaverton, OR 97005
(503) 644-3535

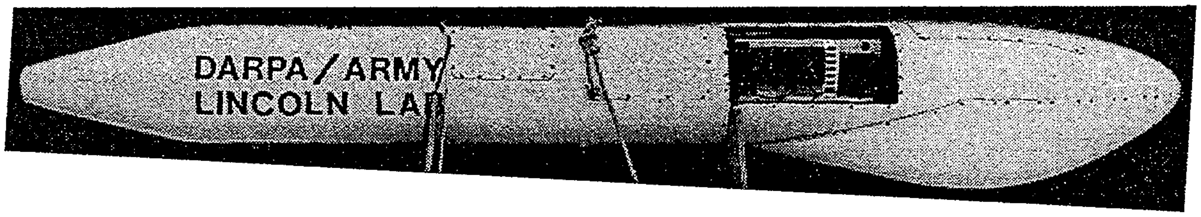
TriQuint® SEMICONDUCTOR

APPENDIX D

Unmanned Air Vehicle

The SAR processor developed for RASSP must be capable of operation within a medium range unmanned air-vehicle (UAV). Processor hardware on-board the UAV used to derive the form-factor constraints Section 2.3 is shown in Figure D.1, with associated dimensional diagrams shown in Figure D.2 (units are inches). A representative processor chassis is shown in Figure D.3.

SIDE VIEW



TOP VIEW

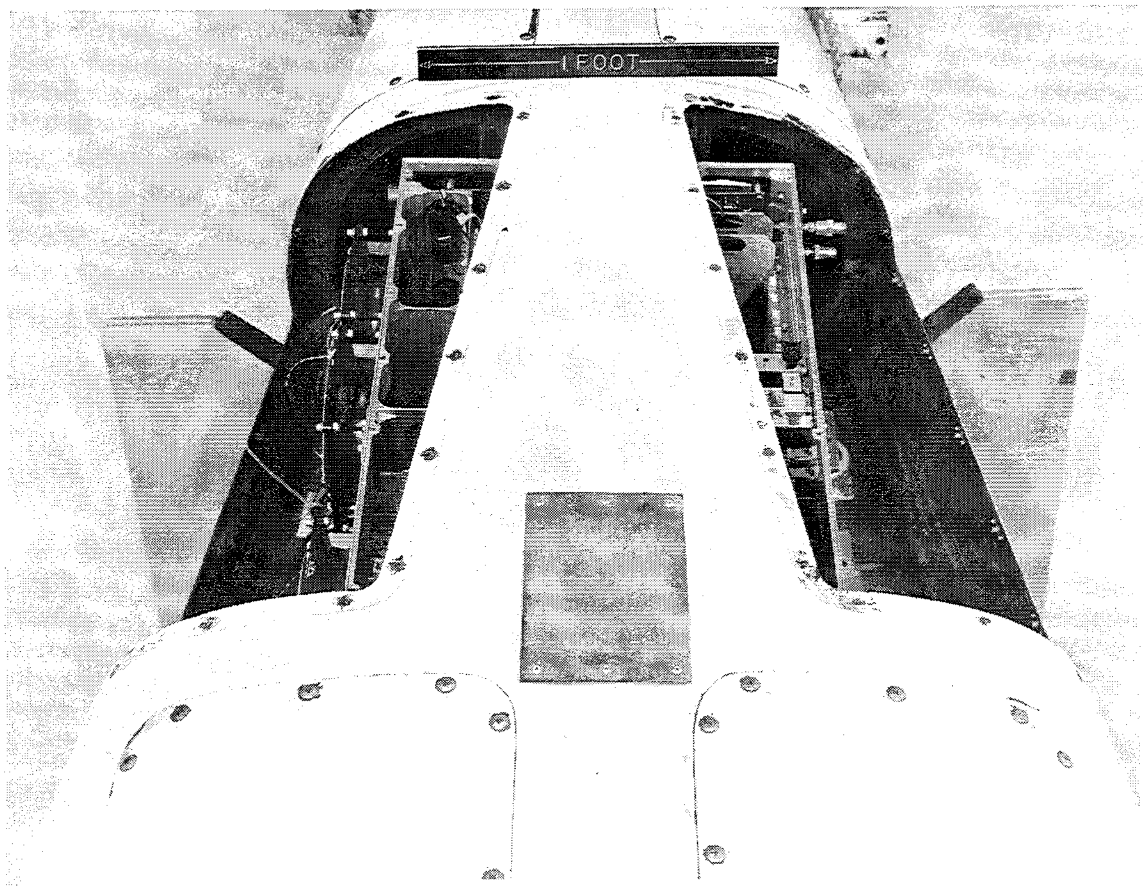


Figure D.1. UAV Photographs.

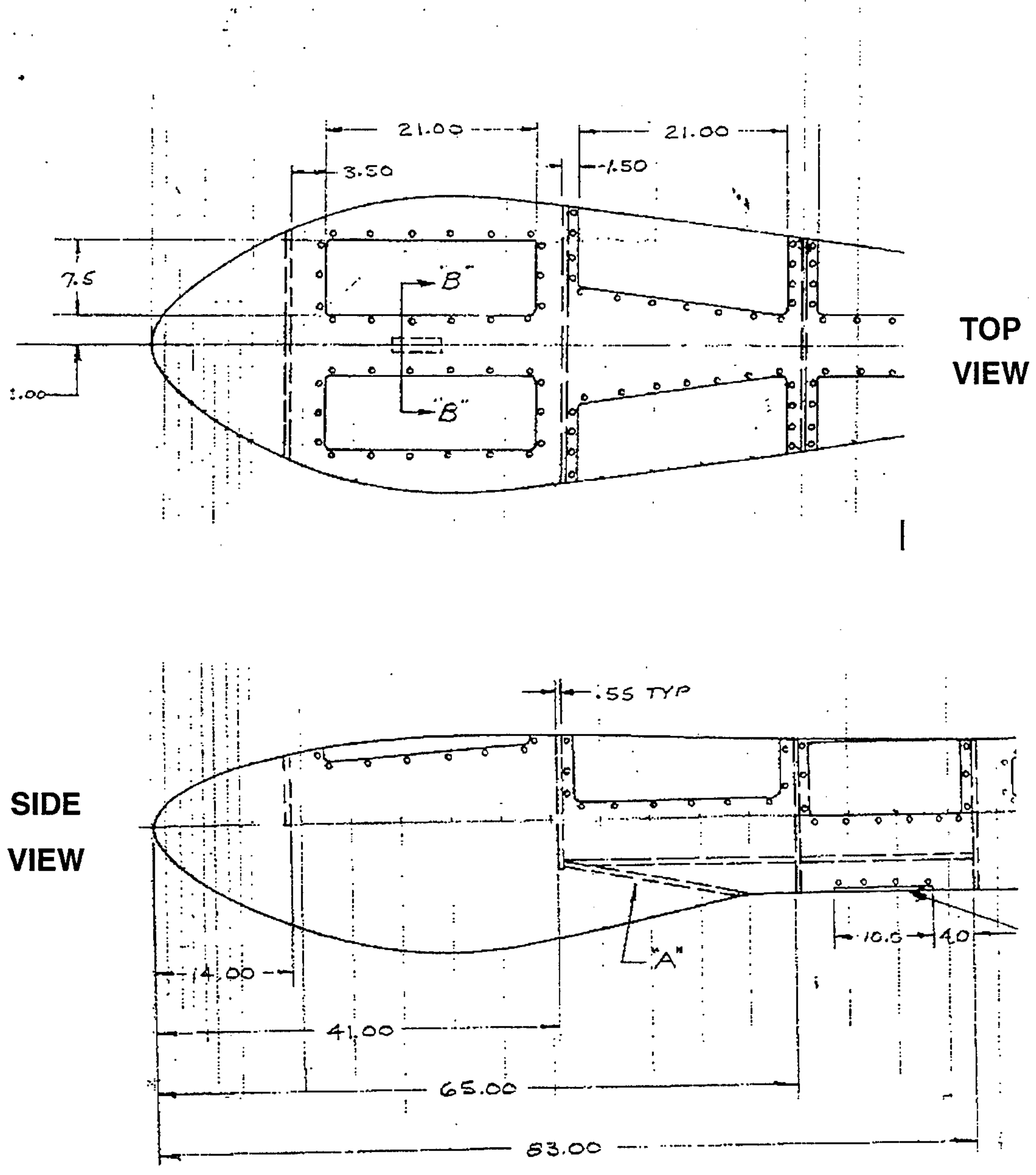


Figure D.2. UAV Drawings

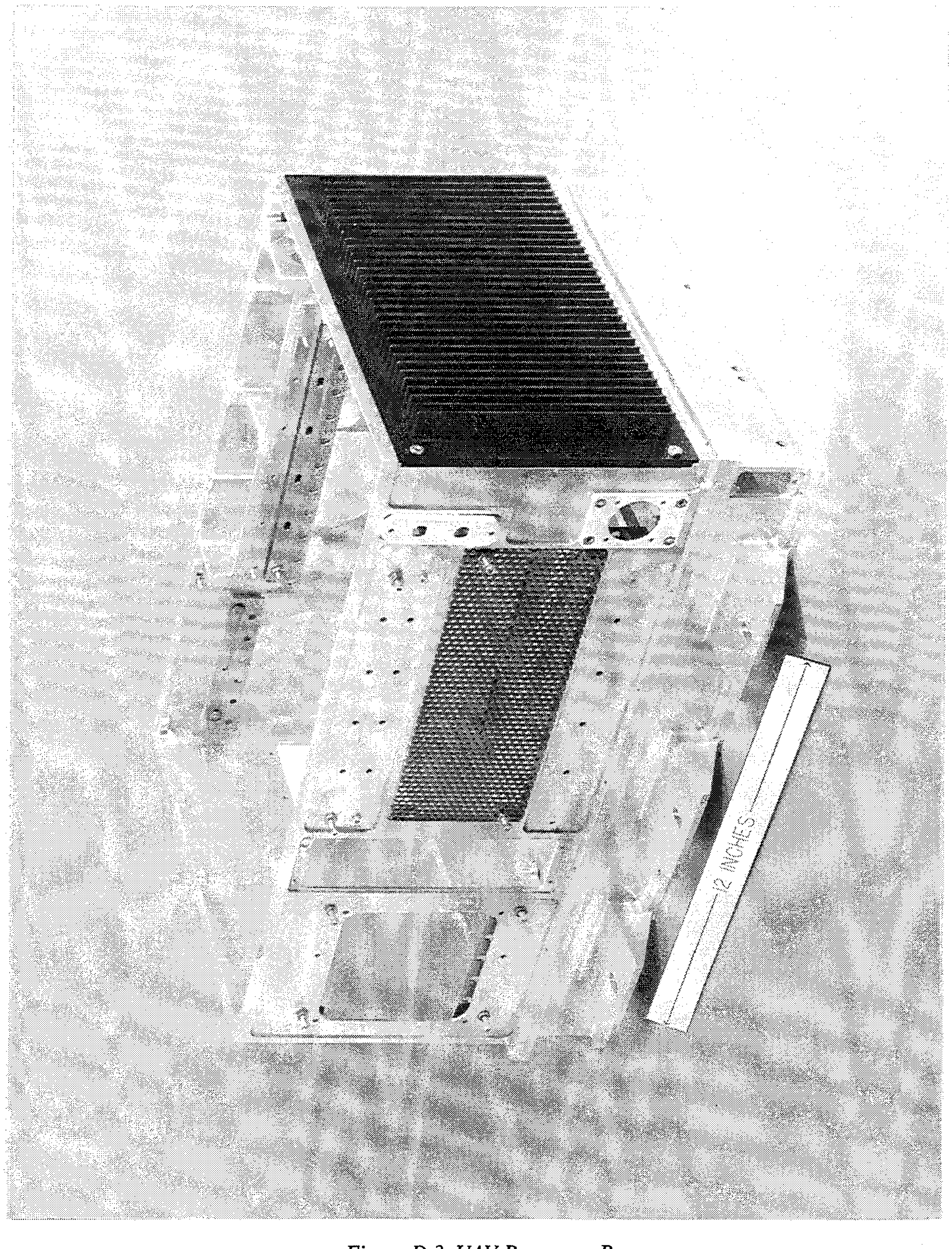


Figure D.3. UAV Processor Box

GLOSSARY

Application Thread	An application chosen as the vehicle for one or more six-month benchmark execution cycles. The first series of benchmarks is based on a real-time processor for synthetic aperture radar imaging.
Benchmark Cycle	A nominal six-month long period during which the Developer applies the current RASSP process to develop an application and meet the requirements defined in a Benchmark Technical Description.
Benchmark Technical Description	The BTD is a document and supporting technical information which defines each benchmark including system requirements, deliverables, and allowable duration.
COCOMO	A well-documented parametric cost estimation tool for software efforts. Many computer programs which implement different versions of the COCOMO equations are available.
Data Source/Sink	The Data Source/Sink is a VME-based turn-key system which shall be delivered to the Developer by the Benchmarker for use in supplying real-time data to RASSP hardware test article, and capturing real-time data produced by the RASSP hardware test article.
PRICE	A suite of parametric cost estimation computer programs from Martin Marietta. The product line presently covers software and software life cycle (PRICE S), microcircuits and electronic assemblies (PRICE M), hardware systems (PRICE H) and hardware life cycle (PRICE HL).
Scalability	As applied to hardware and software architectures is the property of being expandable to address new requirements without substantially changing the design or the existing components.
SEER	A suite of parametric cost estimation computer programs from Galorath Associates. The product line presently consists of a software sizing model (SEER-SSM), software estimation model (SEER-SEM), integrated circuit model (SEER-IC), hardware estimation model (SEER-H) and hardware life cycle model (SEER-HLC).
Virtual prototyping	The process of simulating all applicable levels of hardware functionality (whether behavioral or register-transfer level) in a hardware description language such as VHDL.

ACRONYMS

A/D	Analog to Digital Converter
ADTS	Advanced Detection Technology Sensor
API	Application Programming Interface
ARCM	Application Requirement Complexity Metric
ARPA	Advanced Research Projects Agency
ASIC	Application Specific Integrated Circuit
ATR	Automatic Target Recognition
BECL	Benchmark Execution Check List
BTD	Benchmark Technical Description
COCOMO	Constructive Cost Model
COTS	Commercial Off-The-Shelf
CSC	Computer Software Component
CSCI	Computer Software Configuration Item
CSU	Computer Software Unit
DFT	Discrete Fourier Transform
DSP	Digital Signal Processor
EBS	Electronic Breakdown Structure
EOF	End of File
FFT	Fast Fourier Transform
FIR	Finite Impulse Response
FPGA	Field-Programmable Gate Array
GOPS	Giga-Operations per Second
GPS	Global Positioning System
HCM	Hardware Complexity Metric
HDL	Hardware Definition Language
HOL	Higher Order Language
IF	Intermediate Frequency
IMU	Inertial Measurement Unit
INU	Inertial Navigation Unit
I/Q	In-phase and Quadrature

LFM	Linear Frequency Modulation
LL	Lincoln Laboratory
LRU	Line Replacement Unit
LSB	Least Significant Bit
Mbps	Megabits per second
MB	Megabyte
MCM	Multi-Chip Module
MDL	Metric Deliverable List
MFLOPS	Millions of Floating Point Operations per Second
MIPS	Millions of Instructions per Second
MOPS	Millions of Operations per Second
MSB	Most Significant Bit
MW	Megaword
PAL	Programmable Array Logic
PCB	Printed Circuit Board
PLD	Programmable Logic Device
PME	Prime Mission Equipment
PRF	Pulse Repetition Frequency
PRI	Pulse Repetition Interval
PRICE	Parametric Review of Information for Costing and Evaluation
PSE	Peculiar Support Equipment
R^4	Range to the fourth power
RASSP	Rapid Prototyping Application-Specific Signal Processors
RCS	Radar Cross Section
REVIC	Revised Intermediate COCOMO
SAR	Synthetic Aperture Radar
SEER	System Evaluation and Estimation of Resources
SEI	Software Engineering Institute
SEM-E	Standard Electronic Module, Format E
SNR	Signal-to-Noise Ratio
UAV	Unmanned Air Vehicle
VME	Versa Module Europe
WBS	Work Breakdown Structure

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY (*Leave blank*)	2. REPORT DATE 25 January 1995	3. REPORT TYPE AND DATES COVERED Project Report

4. TITLE AND SUBTITLE RASSP Benchmark 1 Technical Description	5. FUNDING NUMBERS C — F19628-95-C-0002 PE — 63739E PR — 394
6. AUTHOR(S) Brian W. Zuerndorfer, et al.	
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Lincoln Laboratory, MIT P.O. Box 73 Lexington, MA 02173-9108	8. PERFORMING ORGANIZATION REPORT NUMBER PR-RASSP-1 (Rev. 1)
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES) ARPA/ESTO 3701 N. Fairfax Dr. Arlington, VA 22203	10. SPONSORING/MONITORING AGENCY REPORT NUMBER ESC-TR-95-001
11. SUPPLEMENTARY NOTES None	
12a. DISTRIBUTION/AVAILABILITY STATEMENT Approved for public release; distribution is unlimited.	12b. DISTRIBUTION CODE

13. ABSTRACT (*Maximum 200 words*)

This report describes the first in a series of application problems which are intended to measure the performance of a process for rapid prototyping of embedded digital signal processors. The rapid prototyping process is being developed for the ARPA/Tri-Services Rapid Prototyping of Application Specific Signal Processors (RASSP) program. The first application problem is to develop a virtual prototype for a real-time digital signal processor capable of forming images from high-resolution synthetic aperture radar data. Details of the application are provided along with design constraints and optimization requirements for the processor. The report also describes product and process metrics which are to be collected to derive measures of process and product performance. The application problem and associated performance metrics comprise what is termed a Benchmark Technical Description.

14. SUBJECT TERMS

design automation	RASSP	rapid prototyping	digital design
synthetic aperture radar	process metrics	parametric cost estimation	digital signal processing
benchmark	VHDL	virtual prototype	

15. NUMBER OF PAGES
170

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT Unclassified	18. SECURITY CLASSIFICATION OF THIS PAGE Unclassified	19. SECURITY CLASSIFICATION OF ABSTRACT Unclassified	20. LIMITATION OF ABSTRACT